

Dr. Dobb's Journal of

#114 APRIL 1986
\$2.95 (3.95 CANADA)

Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Programming in
LISP and PROLOG

An Expert at Life

The Perils of
Protected Mode

I/O Redirection
for the Shell

ANNUAL AI ISSUE

A) In the first line of your sonnet which reads, "Shall I compare thee to a summer's day," would not "a spring day" do as well?
- It wouldn't scan.

A) How about "a winter's day?" That would scan all right.
- Yes, but nobody wants to be compared to a winter's day.

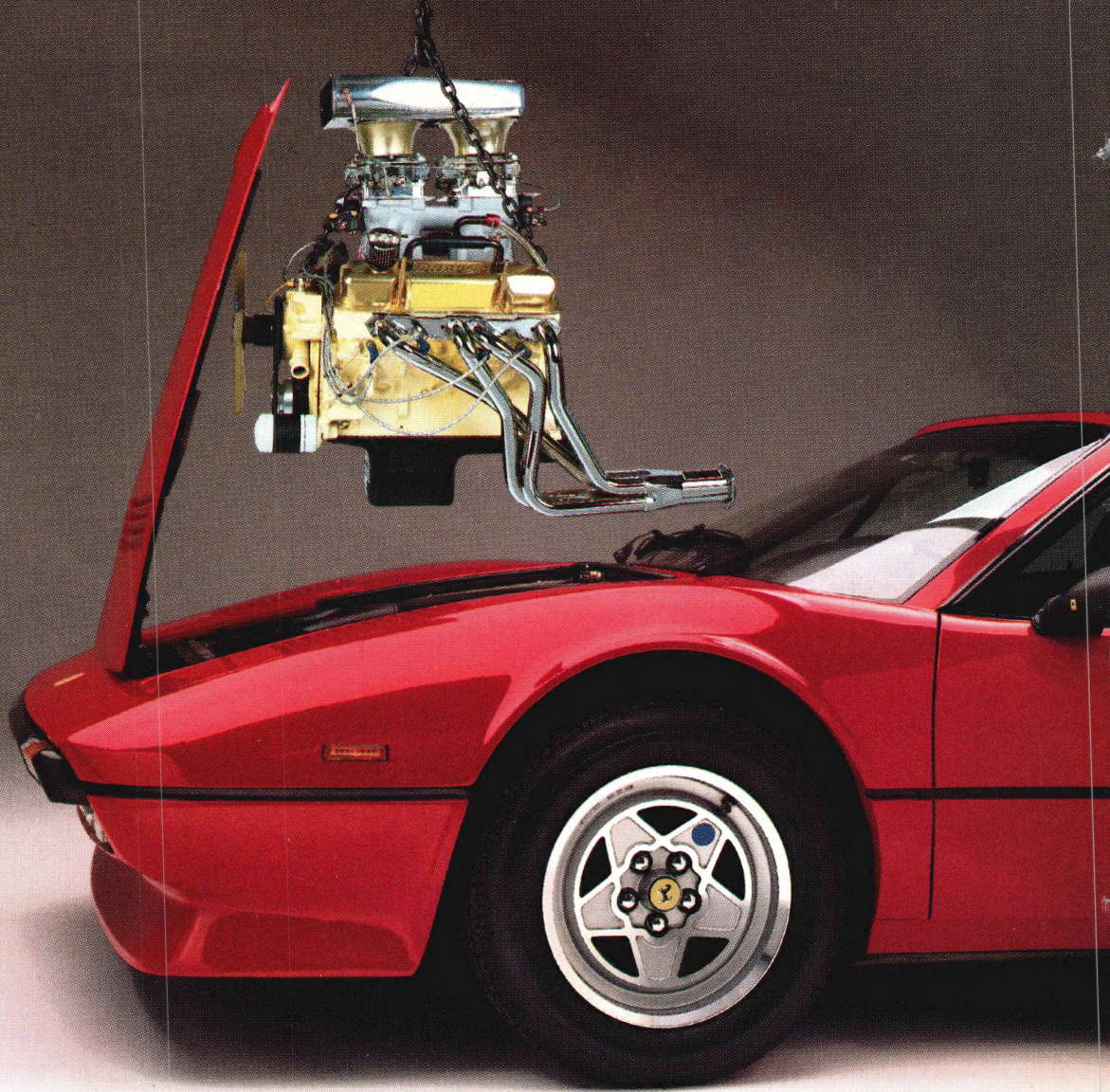
A) Would you say Mr. Pickwick reminded you of Christmas?
- In a way.

A) Yet Christmas is a winter's day, and I do not think Mr. Pickwick would mind the comparison.

- I don't think you're serious. By a winter's day one means a typical winter's day, not a special one like Christmas.

A) _





AT™ Pfantasies for your PC or XT.™

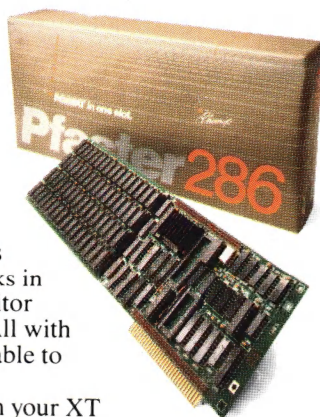
Want better speed and memory on your PC or XT without buying an AT?

You've got it!

Phoenix's new Pfaster™286 co-processor board turns your PC or XT into a high-speed engine 60 percent faster than an AT. Three times faster than an XT. It even supports PCs with third-party hard disks. But that's only the beginning.

You can handle spreadsheets and programs you never thought possible. Set up RAM disks in both 8088 and 80286 memory for linkage editor overlays or super-high-speed disk caching. All with Pfaster286's 1mb of standard RAM, expandable to 2mb, and dual-mode design.

You can develop 8086/186/286 software on your XT faster. Execute 95 percent of the application packages that run on the AT, excluding those that require fancy I/O capabilities your PC or XT hardware just isn't designed to handle. Queue multi-copy, multi-format print jobs for spooling. Or, switch to native 8088 mode to handle



hardware-dependent programs and back again without rebooting. All with Pfaster286's compatible ROM software. And, Pfaster286 does the job unintrusively! No motherboard to exchange. No wires to solder. No chips to pull. Just plug it into a standard card slot, and type the magic word, "PFAST."

If you really didn't want an AT in the first place, just what it could do for you, call or write: Phoenix Computer Products Corp., 320 Norwood Park South, Norwood, MA 02062; (800) 344-7200. In Massachusetts, 617-762-5030.

Programmers' Pfantasies™
by

Phoenix

XT and AT are trademarks of International Business Machines Corporation. Pfaster286 and Programmers' Pfantasies are trademarks of Phoenix Computer Products Corporation. For the Ferrari aficionado: yes, we know this is a rear engine car. We are showing the addition of a second engine to symbolize how Pfaster can be added to your PC or XT to increase performance.

Circle no. 91 on reader service card.



Optotech, Inc.

The 5¼ inch Optical Disk Drive Is Here!

Optical Disk Drive 5984

- 200 megabytes on a removable cartridge.
- Fast access read and write.
- For PCs and minis.
- Extensive interface software.
- Available Now.

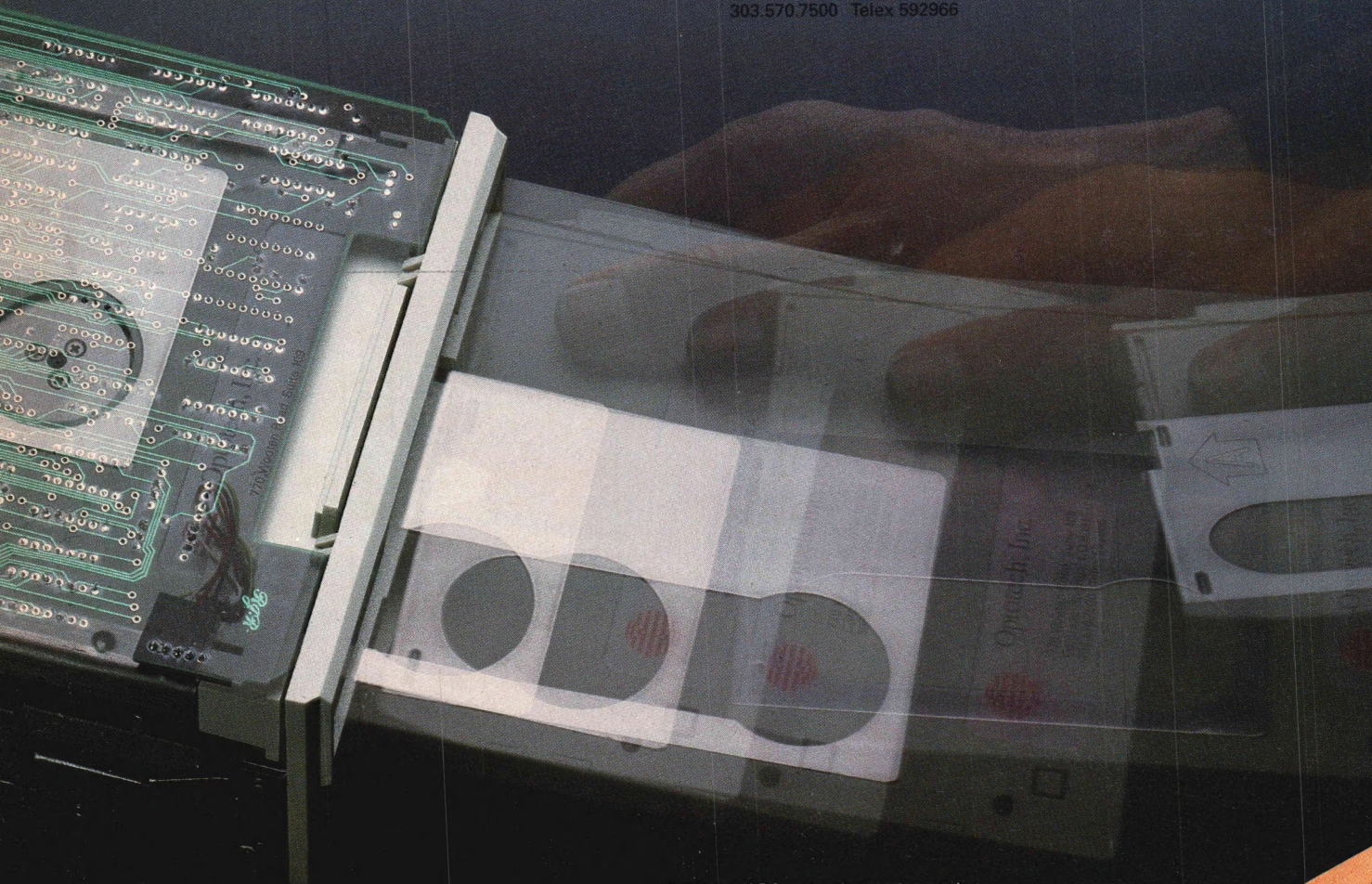
The Preferred Solution For:

- Database—large, portable, indelible and updatable
- Online Mass Storage—integral backup
- Imaging—capacity with removability



Optotech, Inc.

770 Wooten Road
Colorado Springs, CO 80915
303.570.7500 Telex 592966



PERFORMANCE PACKAGE

Blaise Computing Inc. introduces the PERFORMANCE PACKAGE™ for Turbo Pascal programmers.

TURBO PASCAL

Turbo ASYNCH™

With Turbo ASYNCH, you can be in constant touch with the world without ever leaving the console. Rapid transit at its best. Turbo ASYNCH is designed to let you incorporate asynchronous communication capabilities into your Turbo Pascal application programs, and it will drive any asynchronous device via the RS232 ports, like printers, plotters, modems or even other computers. Turbo ASYNCH is fast, accurate and lives up to its specs. Features include...

- ◆ Initialization of the COM ports allowing you to set all transmission options.
- ◆ Interrupt processing.
- ◆ Data transfer between circular queues and communications ports.
- ◆ Simultaneous buffered input and output to both COM ports.
- ◆ Transmission speeds up to 9600 Baud.
- ◆ Input and output queues as large as you wish.
- ◆ XON/XOFF protocol.

The underlying functions of Turbo ASYNCH are carefully crafted in assembler for efficiency, and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The interface to the assembler routines is written in Turbo Pascal.

The Turbo Pascal PERFORMANCE PACKAGE™ is for the serious Turbo Pascal programmer who wants quality tools to develop applications. Every system comes with a comprehensive User Reference Manual, all source code and useful sample programs. They require an IBM PC or compatible, utilizing MS-DOS version 2.0 or later. There are no royalties for incorporating PERFORMANCE PACKAGE functions into your applications.

Turbo POWER TOOLS and Turbo ASYNCH sell for \$99.95 each, and they may be ordered directly from Blaise Computing Inc. To order, call (415) 540-5441.

BLAISE COMPUTING INC.

BLAISE

NEW!

Turbo POWER TOOLS™

Turbo POWER TOOLS is a sleek new series of procedures designed specifically to complement Turbo Pascal on IBM and compatible computers. Every component in Turbo POWER TOOLS is precision engineered to give you fluid and responsive handling, with all the options you need packed into its clean lines. High performance and full instrumentation, including...

- ◆ Extensive string handling to complement the powerful Turbo Pascal functions.
- ◆ Screen support and window management, giving you fast direct access to the screen without using BIOS calls.
- ◆ Access to BIOS and DOS services, including DOS 3.0 and the IBM AT.
- ◆ Full program control by allowing you to execute any other program from within your Turbo Pascal application.
- ◆ Interrupt service routines written entirely in Turbo Pascal. Assembly code is not required even to service hardware interrupts like the keyboard or clock.

Using Turbo POWER TOOLS, you can now "filter" the keyboard or even DOS, and create your own "sidekickable" applications.

YES, send me the Best for the Best! Enclosed is _____ for
☐ Turbo ASYNCH ☐ Turbo POWER TOOLS. (CA residents add
6½% Sales Tax. All orders add \$6.00 for shipping.)

Name: _____ Phone: _____

Shipping Address: _____ State: _____ Zip: _____

City: _____ Exp. Date: _____

VISA or MC #: _____

Turbo Pascal is a trademark of Borland International. Turbo POWER TOOLS, Turbo ASYNCH and PERFORMANCE PACKAGE are trademarks of Blaise Computing Inc. IBM is a registered trademark of International Business Machines Corporation. MS-DOS is a trademark of Microsoft Corporation.

Watch us!

- ◆ 2034 BLAKE STREET
- ◆ BERKELEY, CA 94704
- ◆ (415) 540-5441

Dr. Dobb's Journal of

Software Tools

ARTICLES

**Programming
in LISP and
PROLOG** ▶**An Expert at
Life** ▶**AI: BRIE—The Boca Raton Inference Engine** **24***by Robert Jay Brown III*

An exploration of artificial intelligence techniques, using LISP, PROLOG, and Expert-2.

AI: A Cellular Automation in Expert-2 **42***by Jack Park*

Jack visited our pages two years ago with an expert system for predicting the weather. This little game could teach even more about AI tools.

AI: Modeling a System in PROLOG **46***by Sheldon D. Softky*

PROLOG may be the language of choice for some very practical tasks says the author.

MODULA-2: A 68000 Cross Assembler—Part 1 **52***by Brian R. Anderson*

The first installment of this Modula-2 source code assembler for the 68000 supplies the definition modules and data-flow diagrams. The implementation modules will follow.

MATH: A Variable Metric Minimizer **84***by Joe Marasco*

The source code listing for Marasco's arbitrary-function minimizer continued from last month.

COLUMNS

**I/O redirection
for the Shell** ▶**Report on
Microsoft's
Macro
Assembler 4.0** ▶**C CHEST: Redirection—The /dev Directory, SWITCHAR, and Touch** **18***by Allen Holub*

Our C expert tackles I/O redirection for his shell program, switching the switch character under DOS 3.x, and a bug in *mk*, and adds a nice touch to the make utility.

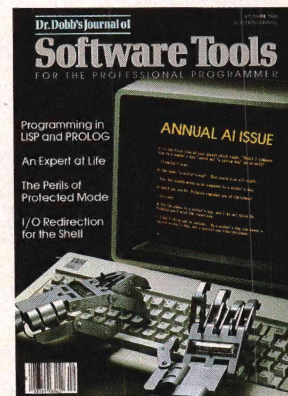
16-BIT SOFTWARE TOOLBOX: The Perils of Protected Mode **116***by Ray Duncan*

With thousands of disks in his customers' hands, Ray learns to his dismay that IBM didn't really mean it about interrupt 0FFH being available. Also: how to use the LaserJet with WordStar, Ray's opinion of Microsoft's Macro Assembler 4.0, 68000 and 8086 tricks, and an MS DOS tail routine.

FORUM

EDITORIAL: Charm **6**
*by Michael Swaine***LETTERS:** Parity **8**
*by our readers***CARTOON:** Strangeness **8**
*by Rand Renfro***DDJ ON LINE:** What's up **14**
and how to get it

PROGRAMMERS' SERVICES

OF INTEREST: New **120**
products of interest
to programmers**ADVERTISER INDEX:** **128**
Where to find
that ad**How to
navigate the
DDJ Forum** ▶

About the Cover

The Turing test is Alan Turing's mythical experiment in which the interrogator tries to determine whether the respondent is human or machine strictly on the basis of the respondent's answers to questions. Paul Ambrose, who created the robot hands and did the cover photography, relaxed the restriction that the interrogator be human. The dialogue is Turing's.

This Issue

Last year's AI issue, which focused on PROLOG, was so popular that we decided to make it an annual thing. This year we've widened the scope to include LISP and Expert-2. The focus is, as always, on providing useful techniques and code.

Next Issue

Some of the best programming acts play to an audience of one because the programmer never learned the difference between a nifty box of tricks and a useful tool that others can understand and use. In our user-interface design issue we'll talk to Jef Raskin, the originator of the Macintosh project, about designing software for people. We'll also turn Jim Edlin loose on a well-known software product; Jim says he's going to use Dan Bricklin's Demo Program to redesign this product's user interface.

It's amazing what you can reveal when you strip.

Introducing a shape that's about to turn on an entire industry.

The Softstrip™ data strip. From Cauzin.

This new technology allows text, graphics, and data to be encoded on a strip of paper, then easily entered into

your computer using a scanning device called the Cauzin Softstrip™ System Reader.

Creating a simple, reliable and cost efficient way to distribute and retrieve information.

Softstrip data strips, like those you see here, can contain anything that can be put on magnetic disks.

Facts. Figures. Software programs.

Video games. Product demonstrations.

Sheet music.



The Cauzin Softstrip System Reader is now compatible with the IBM PC, Apple II and Macintosh.

A single strip can hold up to 5500 bytes of encoded data.

It can stand up to wrinkles, scratches, ink marks, even coffee stains.

And it can be entered into your computer with a higher degree of reliability than most magnetic media.

Simply by plugging the Cauzin Reader into your serial or cassette port and placing it over the strip.

The reader scans the strip, converts it to computer code, and feeds it into any standard communication interface.

Because strips are so easy to generate, most of your favorite magazines and books will soon be using them in addition to long lists of program code.

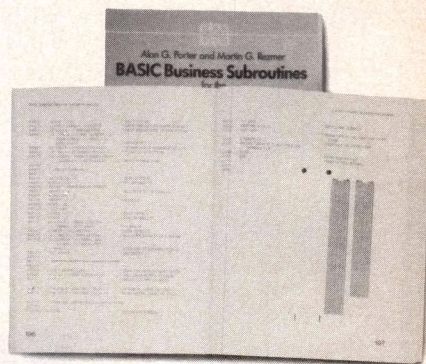
And you'll be able to enter programs without typing a single line.

There is also software for you to generate your own strips.

Letting you send everything from correspondence to business information using our new technology.

Find out how much you can reveal by stripping. Just take this ad to your computer dealer for a demonstration of the Cauzin Softstrip System Reader.

Or for more information and the name of the dealer nearest you, call Cauzin at 1-800-533-7323. In Connecticut, call 573-0150.



Soon everyone will be stripping as data strips appear in popular magazines, computer books and text books.



Cauzin Systems, Inc.
835 South Main St., Waterbury, CT 06706

Apple® and Macintosh® are registered trademarks of Apple Computer Inc., Apple® is a registered trademark of Apple Records, Inc., Softstrip® and the Softstrip™ System Reader are trademarks of Cauzin Systems, Inc., IBM® is a registered trademark of IBM, Inc.

Circle no. 226 on reader service card.

MAKING THE GOOD LIFE EVEN BETTER

Someone once said that there is nothing new under the sun. Wouldn't life be boring if that were indeed true? The data strips on the right contain the program described in the article "The Game of Life in Expert-2", by Jack Park, which appears in this issue. It's a prime example of how something, in this case the game of LIFE itself, can, indeed be improved.

The game of LIFE was invented years ago by John Horton Conway. Over the years, the game has evolved into a popular cerebral exercise for programmers and math majors alike. At first the game was played on graph paper, but the advent of modern technology moved it to the computer which plays the game thousands of times faster. Now millions of computer enthusiasts are captivated by this devilishly simple, yet marvelously complex quintessential computer diversion.

The rules of the game are quite simple. Imagine that you have an infinite grid of squares, each one being either alive (on) or dead (off). Each square (called a "cell") lives or dies into the next cycle (called a "generation") based on its current state and that of its neighbors. The grid of cells is represented by a graphic display on your computer screen. After setting up an initial configuration of living and dead cells, you start the simulation. The patterns will change on the screen as cells live and die.

Mr. Park's improvement on the theme is interesting because of his approach. Instead of writing a traditional program for the simulation, he has created an array of intelligent cells using an inference engine written in Expert-2, a superset of FORTH.

Read in the data strips, following the directions that came with your Cauzin reader. You'll need the Expert-2 programming environment to operate this program. Refer to Mr. Park's article in this issue for operating instructions.

Reprinted with permission of Dr. Dobb's Journal.

StripWare Library No. 202

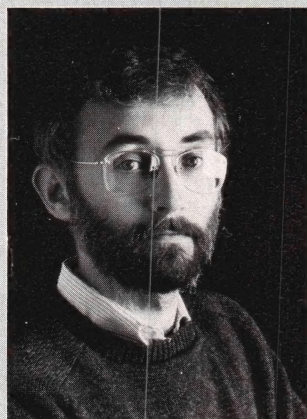
1

2

3

Softstrip

EDITORIAL



Charles Dickens printed on the front page of his magazine, *Household Words*, a public explanation of the breakup of his marriage. He told his proper Victorian readers that he and Catherine Dickens were "wonderfully unsuited to one another" and that "it would be better for her to go away and live apart." Magazines are different now. Living with someone else is, in some ways, no different: Now, as then, the bedrock on which lives are built is occasionally faulty. Like plates of the earth's crust, cohabitants sometimes rub each other the wrong way until something cracks, erupts, falls away.

Memory-resident utilities under MS DOS are also subject to the tectonics of cohabitation, one moment displaying a surface of smoothly overlapping plates or proper Victorian tilings, the next moment falling apart at the fault lines. The DOS terrain needs a seismologist, or the DOS household needs a marriage counselor, depending on where in these shifting metaphors you come to rest.

Remember integrated software? (There is a connection here, I assure you.) Recent trends (evidenced at a utility fair in San Francisco sponsored by 800 Software and *Computer Currents* magazine) suggest that users soon may be offered "integrated software" in a new form: bundled memory-resident utilities. Traditional integrated packages no longer enjoy the vogue they once did, in part because they forced consumers and developers either to accept inadequate tools because they were inextricably bound to better tools or to reject whole packages.

Small utility programs, as contrasted with monolithic integrated packages, have many virtues: They give the user a choice, they provide a

chance for smaller companies to start and succeed, and they encourage a healthier and more competitive market with more differentiable products. Memory-resident programs put a premium once again on memory-efficient design, that is, skillful programming. Bundled memory-resident utilities may or may not limit users' choices, but they still leave it to consumers to choose what products to use. That freedom is beneficial to everyone.

We applaud efforts at agreement on an open TSR (Terminate and Stay Resident, that is, memory-resident) environment and the concept of user-integrable software. As would, we have no doubt, Catherine Dickens.

Michael Swaine

Michael Swaine

Dr. Dobb's Journal of

Software Tools

Editorial

Editor-in-Chief Michael Swaine
Managing Editor Vince Leone
Assistant Editor Sara Noah Buddy
Technical Editor Allen Holub
Contributing Editors Ray Duncan
 Allen Holub

Copy Editor Rhoda Simmons
Electronic Editor Levi Thomas

Production

Production Manager Bob Wynne
Art Director Shelley Rae Doeden
Production Assistant Alida Hinton
Typesetter Jean Aring
Cover Artist Paul Ambrose

Circulation

Fulfillment and
Newsstand Mgr. Stephanie Barber
Subscription Mgr. Maureen Kaminski
Book Marketing Mgr. Jane Sharninghouse
Circulation Assistant Kathleen Shay

Administration

Finance Manager Sandra Dunie
Business Manager Betty Trickett
Accounts Payable Supv. Mayda Lopez-Quintana
Accounts Payable Assts. Denise Giannini
 Kathy Robinson
Billing Coordinator Laura Di Lazzaro
Accountant Marilyn Henry
Adm. Coordinator Kobi Morgan

Advertising

Director
 Shawn Horst (415) 366-3600
Account Managers
 Walter Andrzejewski (617) 868-1524
 Lisa Boudreau (415) 366-3600
 Michele Beaty (317) 875-8093
 Michael Wiener (415) 366-3600
Promotions/Srvcs. Mgr. Anna Kittleson
Advertising Secretary Michelle A. Davie

M&T Publishing, Inc.

Chairman of the Board Otmar Weber
Director C.F. von Quadt
President and Publisher Laird Foshay

Dr. Dobb's Journal (USPS 3076900) is published monthly by M&T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600, Second class postage paid at Palo Alto and at additional entry points.

Article Submissions: Send manuscripts and disk (with article and listings) to the Assistant Editor.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128.

ISSN 0884-5395

Customer Service: For subscription problems call: outside CA 800-321-3333; within CA 619-485-9623 or 566-6974. For book, back issue, or disk order problems call 415-424-1474.

Subscription Rates: \$29.97 per year within the United States. Foreign subscription rates: \$56.97 for airmail, \$46.97 for surface mail. Foreign subscriptions must be pre-paid in U.S. Dollars, drawn on a U.S. Bank. TELEX: 752-351

Foreign Distributor: Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016, (212) 6686-1520 TELEX: 620430 (WUI)

Entire contents copyright © 1986 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.



People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.



The C for Microcomputers

PC-DOS, MS-DOS, CP/M-86, Macintosh, Amiga, Apple II, CP/M-80, Radio Shack, Commodore, XENIX, ROM, and Cross Development systems

MS-DOS, PC-DOS, CP/M-86, XENIX, 8086/80x86 ROM

Manx Aztec C86

"A compiler that has many strengths... quite valuable for serious work"

Computer Language review, February 1985

Great Code: Manx Aztec C86 generates fast executing compact code. The benchmark results below are from a study conducted by Manx. The Dhrystone benchmark (CACM 10/84 27:10 p1018) measures performance for a systems software instruction mix. The results are without register variables. With register variables, Manx, Microsoft, and Mark Williams run proportionately faster. Lattice and Computer Innovations show no improvement.

	Execution Time	Code Size	Compile/Link Time
Dhrystone Benchmark			
Manx Aztec C86 3.3	34 secs	5,760	93 secs
Microsoft C 3.0	34 secs	7,146	119 secs
Optimized C86 2.20J	53 secs	11,009	172 secs
Mark Williams 2.0	56 secs	12,980	113 secs
Lattice 2.14	89 secs	20,404	117 secs

Great Features: Manx Aztec C86 is bundled with a powerful array of well documented productivity tools, library routines and features.

Optimized C compiler	Symbolic Debugger
AS86 Macro Assembler	LN86 Overlay Linker
80186/80286 Support	Librarian
8087/80287 Support Lib	Profiler
Extensive UNIX Library	DOS, Screen, & Graphics Lib
Large Memory Model	Intel Object Option
Z (vi) Source Editor -c	CP/M-86 Library -c
ROM Support Package -c	INTEL HEX Utility -c
Library Source Code -c	Mixed memory models -c
MAKE, DIFF, and GREP -c	Source Debugger -c
One year of updates -c	CP/M-86 Library -c

Manx offers two commercial development systems, Aztec C86-c and Aztec C86-d. Items marked -c are special features of the Aztec C86-c system.

Aztec C86-c Commercial System	\$499
Aztec C86-d Developer's System	\$299
Aztec C86-p Personal System	\$199
Aztec C86-a Apprentice System	\$49

All systems are upgradable by paying the difference in price plus \$10.

Third Party Software: There are a number of high quality support packages for Manx Aztec C86 for screen management, graphics, database management, and software development.

C-tree \$395	Greenleaf \$185
PHACT \$250	PC-lint \$98
HALO \$250	Amber Windows \$59
PRE-C \$395	Windows for C \$195
WindScreen \$149	FirstTime \$295
SunScreen \$99	C Util Lib \$185
PANEL \$295	Plink-86 \$395

MACINTOSH, AMIGA, XENIX, CP/M-68K, 68k ROM

Manx Aztec C68k

"Library handling is very flexible... documentation is excellent... the shell a pleasure to work in... blows away the competition for pure compile speed... an excellent effort."

Computer Language review, April 1985

Aztec C68k is the most widely used commercial C compiler for the Macintosh. Its quality, performance, and completeness place Manx Aztec C68k in a position beyond comparison. It is available in several upgradable versions.

Optimized C	Creates Clickable Applications
Macro Assembler	Mouse Enhanced SHELL
Overlay Linker	Easy Access to Mac Toolbox
Resource Compiler	UNIX Library Functions
Debuggers	Terminal Emulator (Source)
Librarian	Clear Detailed Documentation
Source Editor	C-Stuff Library
MacRam Disk -c	UniTools (vi, make, diff, grep) -c
Library Source -c	One Year of Updates -c

Items marked -c are available only in the Manx Aztec C86-c system. Other features are in both the Aztec C86-d and Aztec C86-c systems.

Aztec C68k-c Commercial System	\$499
Aztec C68d-d Developer's System	\$299
Aztec C68k-p Personal System	\$199
C-tree database (source)	\$399
AMIGA, CP/M-68k, 68k UNIX	call

Apple II, Commodore, 65xx, 65C02 ROM

Manx Aztec C65

"The AZTEC C system is one of the finest software packages I have seen"

NIBBLE review, July 1984

A vast amount of business, consumer, and educational software is implemented in Manx Aztec C65. The quality and comprehensiveness of this system is competitive with 16 bit C systems. The system includes a full optimized C compiler, 6502 assembler, linkage editor, UNIX library, screen and graphics libraries, shell, and much more. The Apple II version runs under DOS 3.3, and ProDOS, Cross versions are available.

The Aztec C65-c/128 Commodore system runs under the C128 CP/M environment and generates programs for the C64, C128, and CP/M environments. Call for prices and availability of Apprentice, Personal and Developer versions for the Commodore 64 and 128 machines.

Aztec C65-c ProDOS & DOS 3.3	\$399
Aztec C65-d Apple DOS 3.3	\$199
Aztec C65-p Apple Personal system	\$99
Aztec C65-a for learning C	\$49
Aztec C65-c/128 C64, C128, CP/M	\$399

Distribution of Manx Aztec C

In the USA, Manx Software Systems is the sole and exclusive distributor of Aztec C. Any telephone or mail order sales other than through Manx are unauthorized.

Manx Cross Development Systems

Cross developed programs are edited, compiled, assembled, and linked on one machine (the HOST) and transferred to another machine (the TARGET) for execution. This method is useful where the target machine is slower or more limited than the HOST. Manx cross compilers are used heavily to develop software for business, consumer, scientific, industrial, research, and educational applications.

HOSTS: VAX UNIX (\$3000), PDP-11 UNIX (\$2000), MS-DOS (\$750), CP/M (\$750), MACINTOSH (\$750), CP/M-68k (\$750), XENIX (\$750).

TARGETS: MS-DOS, CP/M-86, Macintosh, CP/M-68k, CP/M-80, TRS-80 3 & 4, Apple II, Commodore C64, 8086/80x86 ROM, 68xxx ROM, 8080/8085/Z80 ROM, 65xx ROM.

The first TARGET is included in the price of the HOST system. Additional TARGETS are \$300 to \$500 (non VAX) or \$1000 (VAX).

Call Manx for information on cross development to the 68000, 65816, Amiga, C128, CP/M-68k, VRTX, and others.

CP/M, Radio Shack, 8080/8085/Z80 ROM

Manx Aztec CII

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen."

80-Micro, December, 1984, John B. Harrell III

Aztec C II-c (CP/M & ROM)	\$349
Aztec C II-d (CP/M)	\$199
C-tree database (source)	\$399
Aztec C80-c (TRS-80 3 & 4)	\$299
Aztec C80-d (TRS-80 3 & 4)	\$199

How To Become an Aztec C User

To become an Aztec C user call 1-800-221-0440 or call 1-800-832-9273 (800-TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers. Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

How To Get More Information

To get more information on Manx Aztec C and related products, call 1-800-221-0440, or 201-530-7997, or write to Manx Software Systems.

30 Day Guarantee

Any Manx Aztec C development system can be returned within 30 days for a refund if it fails to meet your needs. The only restrictions are that the original purchase must be directly from Manx, shipped within the USA, and the package must be in resalable condition. Returned items must be received by Manx within 30 days. A small restocking fee may be required.

Discounts

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade in" basis for users of competing systems. Call for information.

Circle no. 108 on reader service card.

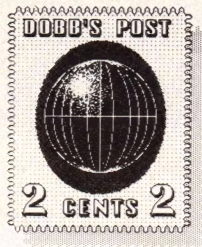
MANX

To order or for information call:

800-221-0440

UNIX is a registered TM of Bell Laboratories. Lattice TM Lattice Inc. C-tree TM Faircom, Inc. PHACT TM PHACT ASSOC. CI Optimizing C86 TM Computer Innovations, Inc. MACINTOSH, APPLE TM APPLE, INC. Pre-C, PLINK 86 TM PHOENIX, HALO TM Media Cybernetics, Inc. C-tree, PC-lint TM GIMPLE Software, WindScreen, SunScreen TM SunSoft, PANEL TM Roundhill Computer Systems Ltd. WINDOWS FOR C TM Creative Solutions, XENIX, MS TM MICROSOFT INC. CP/M TM DRI, AMIGA, C64, C128 TM COMMODORE INT.

LETTERS



Columns

Dear DDJ,

Hal Hardenbergh's Viewpoint column "Inefficient C" (DDJ, January 1986) correctly points out the obvious—that regardless of the arguments high-level language (HLL) advocates bring to bear, HLL software for microcomputers does not perform as well as good assembly-language software and is consequently less popular with the buying public. His reasoning as to why this is so is valid in my experience (extensive 808x and Z80 programming in both assembly and HLL); however, I think that the ongoing debate about the efficiency of assembly versus HLL is only the focal point of a larger issue, which concerns the attempt to manage the creative process of software development. After all, why has such an acrid debate arisen on a topic about which, as Mr. Hardenbergh has pointed out in *DTACK Grounded*, there can be little doubt (for microcomputers, at least)? Because standard management practices and corporate dynamics make it very difficult to follow the software development approach required to produce good assembly programs.

From a management perspective, software development falls into an odd area somewhere between creative and technical writing. Technical writing

can be outlined, broken into blocks, and implemented by many writers because it is a description of something that already exists in finished form. However, it would be foolish to attempt to do the same with a novel because writing a novel is a creative act. Even if the author could fully outline the novel, no team of writers could make it come together; no matter how well each wrote his or her individual sections, the parts would not cohere. Software development has elements of both sorts of writing in that it requires careful organization yet is essentially a creative process.

It is easy to fall into the trap of viewing software development as only a kind of technical writing. After all, if you ask a novice programmer to write a program that prints out

the squares of the integers less than ten, he will deliver a working program much like the ones written by any other programmer. By extension, programming becomes a basically mechanical task of implementing an already-determined solution, with the creative work completed when the program specification is finished. All too often the specification is developed by people who are not even expert programmers, the perception being that creation and implementation are separate parts of software development.

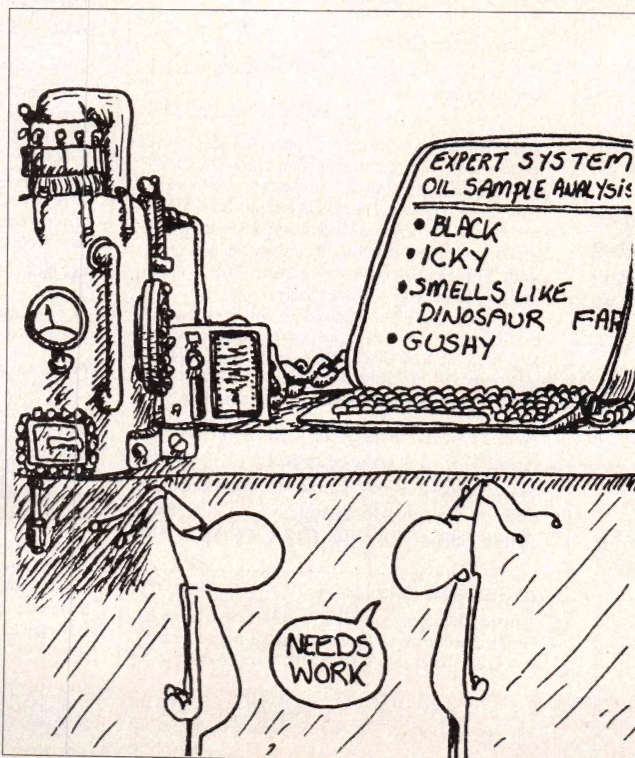
This is a critical misperception. Good microcomputer software results from a development process in which the creative element, the implementation, and the underlying hardware must work together throughout the development process.


The best microcomputer software is written in assembly language under a single unifying vision. Both the vision and assembly language are essential to allow the best matching of the application, via the code and data structures, to the underlying architecture of the computer. As Mr. Hardenbergh indicates, it is this marriage of software and hardware that makes assembly programs superior. Good assembly software can only be implemented so long as the vision can be shared and common elements fully communicated among the entire team. Because communications breakdown is the bottleneck, ideally one and never more than an absolute minimum number of programmers should be involved in any one project.

Given that the best software is written in assembly by a very few programmers, why do so many companies end up using HLL? A number of factors force large and growing software companies into using HLL (and in this industry, there are generally only large, growing, and dying software companies):

1. The need to deal with the inevitable turnover of employees. This virtually mandates HLL because it is difficult to maintain someone else's assembly code. Exacerbating this is the tendency of good software developers to move on to more interesting projects (and companies).

2. The preference for cheap labor. In rapidly growing companies, the tendency is to add a second tier of programmers—often right out of school—who are paid much less





XTC[®] The Ultimate
Programmer's
Editor

When Will It Dawn On You?

XTC[®] is a PC programmer's dream come true. Just ask the thousands of programmers who've already awakened to the extraordinary features of this world class editor, designed for the IBM PC and true compatibles.

Features like XTC's interpretive macro language that lets you code macros on the fly. And fine-grained multitasking so you can run macros in the background while you continue editing.

Of course, XTC also lets you code macros in a high level language. And with the macro compiler you can write macros to check syntax in real time and translate source code from one language to another.

What's more, DOS compilers, linkers and utilities will run inside XTC so you don't have to leave the editor to compile and test run your programs.

If that's not enough, XTC has multiple large windows, 20 text buffers, and a host of other powerful features no other editor can offer.

In brief, you'll find little to compare with the power, flexibility, and advanced features that make XTC outshine the competition every time.

So if you're still dreaming about the ultimate editor, wake up. It could be right before your eyes.

XTC. From Wendin. Only \$99.

WENDIN[®]

BOX 266
CHENEY, WA 99004

© Copyright 1986 Wendin, Inc.

The people who make quality
software tools affordable.

MS is a trademark of Microsoft. PC-DOS is a trademark of IBM. UNIX is a trademark of AT&T. VAX/VMS is a registered trademark of Digital Equipment Corporation.

Wendin and XTC are registered trademarks of Wendin, Inc. PCUNIX, PCVMS, Operating System Toolbox, and Personal Operating System are trademarks of Wendin, Inc.

Ask about our other
products for the IBM PC
and true compatibles.

PCVMS[™]

Multitasking, multiuser version of DEC's powerful VAX/VMS operating system. Runs most MS-DOS programs.

PCUNIX[™]

True multitasking, multiuser operating system similar to AT&T's popular UNIX[®] operating system.

OPERATING SYSTEM TOOLBOX[™]

Complete software construction set that lets you build your own multitasking, multiuser operating systems.

**All products priced at \$99 with
source code included.**

ORDER HOTLINE
(509) 235-8088
(MON.-FRI., 8-5 PACIFIC TIME)



DEALER INQUIRIES WELCOME

Foreign orders inquire about shipping.
Domestic orders add \$5.00/1st item,
\$1.00 each additional item for shipping,
handling, and insurance.
Washington residents add 7.8% sales tax.

Circle no. 112 on reader service card.

LETTERS

(Continued from page 8)

than the original group. These people are truly programmers rather than software developers, rarely have any experience with assembly language (or even with micros), and are often brainwashed with the doctrine of HLL superiority. It takes considerable experience to become an expert assembly programmer, and companies believe they can't afford to let their employees learn on the job.

3. The need to control employees. The same mindset that makes good programmers work all night when the company is developing that first exciting product leads them to follow their own star when they're bored with maintaining that code. Erratic hours and behavior may be fine when the company consists of a programmer and a president in a single room, but they do not sit well with a newly minted department head who's trying to organize his programming staff—and good programmers are a notoriously eccentric bunch.

4. The need to manage projects. As a company grows, it accumulates personnel, even if it is not always clear what these people are hired to do. There is the natural desire of managers to exert more influence by having a larger staff and budget, and there is the equally natural desire to hire more programmers to produce more products and so cause the company to grow faster. Once you have many programmers, you must use them. Consequently, ten or more programmers may end up working on a project that would have been

handled by one or two a year earlier. It is difficult to manage so large a staff when coding in assembly language. (As Mr. Hardenbergh pointed out, however, HLLs make it easier to manage large staffs by obscuring the architecture of the computer and by limiting the tools available to the programmer.) Indeed, if the staff gets much larger, a Unix-based mini and SCCS will be required just to keep chaos at bay. To many this may sound fine, but remember that I am talking only about a single microcomputer product that could likely have been developed by just one or two talented and experienced assembly programmers!

Note that all these factors treat software developers as technical writers—cogs in the machinery that produces a product—rather than as novelists—the source of the product, and the resource around which the company is built. Marketing, finance, management, and so on, certainly must also be good, but they are there to support the product, and it is the quality of the product that will define the company's long-term success.

One example in an article discussing the virtues of Modula-2 (*PC Week*, September 24, 1985), MicroPro explained that it was pleased with Modula-2 because it had enabled 50 people to write Easy, its new low-end word processor, in only nine months. Everyone in the audience who thinks that they and the experienced programmer of their choice could develop that software (apart from a full set of printer drivers, perhaps) in assembly in nine months, raise their hand. (This being *DDJ*, I ex-

pect a lot of hands.) And would anyone care to bet which version would run faster?

The point is not to slight either Modula-2 or MicroPro, but rather to indicate that conventional management inexorably drives corporate software development in such a way that a company can be pleased by the somewhat absurd outcome of getting a low-end product from dozens of highly paid people in nine months. The need for HLLs is merely one result of this process.

In short, the development of good microcomputer software is not amenable to standard management practices. Good software developers are not just implementers, and likewise, good software is not just the implementation of a concept—the concept, the software, and the hardware together form a whole. The current debate about C versus assembly is primarily a reflection of that reality. If software written in C were really as efficient as that written in assembly, then management would have their way, and the issue would be moot. Because managers must manage and because HLLs lend themselves to standard management, a great deal of effort has gone into believing that this is true. Because it isn't, management often ends up wondering why superbly managed projects produce mediocre software. It's not that management techniques aren't applied thoroughly enough—it's a fundamental clash between those techniques and the nature of computers and creativity. This does not mean that the development of good software cannot be managed, scheduled, and

otherwise integrated into the corporate structure; it does mean that innovative management is required to support the special nature of software development.

Michael Abrash

6 Remy Pl.

Newtown, PA 18940

Dear *DDJ*,

In his January column, Ray Duncan states:

"I put the people who upload such programs to public BBSs in the same category as terrorists."

I, too, deplore this conduct. Nonetheless, to compare the loss of a disk of data to the murder of 19 humans is inappropriate.

By the way, in your search for good columnists, may I suggest Hal Hardenbergh? Even though he does not have

the time to write *DTACK Grounded* any longer, he may have time to do a column. His opinions are always interesting, even if misguided. (I have the system he most maligns—I program in C on a Berkeley 4.2 Unix system with a National Semi 16032 CPU—and I still like to read Hal!)

Ivan Strand

145 Del Mar Ave.

Berkeley, CA 94708

We've been hounding Hal Hardenbergh for two years to write for us. Maybe your letter will help. Maybe spelling his name correctly (which we failed to do in January) will help.—ed.

BANKSWAP

Dear *DDJ*,

Albert Woodhull's article, "BANKSWAP" (December 1985), illustrating the use of the CP/M 3 RSX procedure for extending the DRI DDT and SID utilities so they can be used to explore banks other than Bank 1, promises to be useful. I'd like to comment on a few points

Mark Williams

Now the biggest name
in C compilers comes in a size
everybody can afford.

Let's C.[™]
\$75

Introducing Mark Williams' \$75 C compiler. Want to explore C programming for the first time? Or just on your own time? Now you can do it in a big way without spending that way. With Let's C.

This is no little beginner's model. Let's C is a powerful programming tool, packed with all the essentials of the famous Mark Williams C Programming System. The one chosen by Intel, DEC, Wang and thousands of professional programmers. The one that wins the benchmarks and the reviewers' praise:

"(This compiler) has the most professional feel of any package we tested..."—BYTE
"Of all the compilers reviewed, (it) would be my first choice for product development."—David W. Smith, PC WORLD

And now for more big news. Get our revolutionary csd C Source

Debugger for just \$75, too. You can breeze through debugging at the C source level ignoring clunky assembler code.

Affordable, powerful, debuggable. Mark Williams Let's C is the big name C compiler at a price you can handle. Get your hands on it now.

Mark Williams Let's C

- For the IBM-PC and MS-DOS
- Fast compact code plus register variables
- Full Kernighan & Ritchie C and extensions
- Full UNIX[™] compatibility and complete libraries
- Small memory model
- Many powerful utilities including linker, assembler, archiver, cc one-step compiling, egrep, pr, tail, wc
- MicroEMACS full screen editor with source
- Supported by dozens of third party libraries
- Upgradeable to C Programming System for large scale applications development

Let's C Benchmark Done on an IBM-PC/XT, no 8087.
Program: Floating Point from BYTE, August, 1983.

Exec Time in Seconds

Let's C	134.20
MS 3.0	347.45

Use this coupon or charge by calling toll-free:
1-800-MWC-1700. In Ill. call 312-472-6659.

ORDER NOW! 60-DAY MONEY BACK GUARANTEE!

Mark Williams Let's C

Please send me:

_____ copies of Let's C and _____ copies of csd (C Source Debugger) at \$75 each. (Ill. residents add 7% sales tax.)

☐ Check ☐ Money Order ☐ Visa, MasterCard or American Express

Name _____

Address _____

City _____ State _____ Zip _____

Card # _____ Exp. Date _____

Signature _____

 **Mark Williams Company**

1430 West Wrightwood
Chicago, Illinois 60614

LETTERS

(Continued from page 10)

made by Mr. Woodhull in the article.

First, user programs can indeed access memory in alternate banks. Advanced Logic Systems, in BIOS version 3.01B1 and later (also older ones?), provided two new BIOS functions to allow reading and writing to the regular Apple memory. Regular Apple memory is Bank 0 in the CP/M 3 implementation that ALS and DRI came out with for the ALS CP/M Card. These new BIOS functions are numbers 33 (APREAD) and 34 (APWRITE). In the ALS 3.01B2 upgrade and the public domain CP-PLUG 3.02B BIOS derived from the ALS 3.01B2 version, a third additional BIOS function, number 35 (CAPPLE) that allows calls to be made to a Bank 0 (Apple memory) address, was added.

The routine in Table 1, below, can be inserted early in a program to set local jumps to the BIOS for interbank read, write, and (if implemented) call operations. I first saw the routine published by the short-lived and now defunct CP/M Plus Users Group (CP-PLUG). You will find it used in, for example, code for MDM7xx and IMP overlays for the ALS CP/M Card. To use these the Apple Bank 0 address being read, written to, or called to, must be in HL and no remapping is necessary. The byte written to or read from this Apple address must be in (APWRITE) or is returned to (APREAD) the Accumulator.

As an example, to ring Apple's bell one needs to call to Apple II location FF3A. Thus, to do that from CP/M:

```
CALL APINIT
LXI H,FF3AH
CALL CAPPL
RET
```

APINIT: (etc.)

You can also do this using the direct BIOS call convention, which makes use of BDOS function number 50. Use BIOS function numbers 33, 34, and 35 for reads, writes, and calls, respectively.

The second point to be made is illustrated by the simple bell-ringer program. It is under some circumstances possible to execute programs in which code is in a different bank from Bank 1, and in the example shown, the execution happens also to be via 6502 code executed by the Apple 6502 CPU. I've uploaded to a few RCP/M's a library file called AP+BRUN.LBR, which includes a program that allows some Apple II binary executable programs to be converted into a CP/M program that, when the CP/M program is run, loads the binary into Bank 0 and then calls to it to execute it.

In this way, for example, disk macros that the Videx Keyboard Enhancer produces on an Apple II DOS 3.3 disk can be transformed into a CP/M-executable file on an Apple II CP/M disk. Thus they can be downloaded while in the CP/M environment.

Finally, the bank-switching BIOS function (SELMEM) is accessible but one must call it from common memory. The RSX approach to scaling the heights to common memory is one way of doing it; another way is simply to have the program relocate bank-switching code there that maintains relations with the program that is operating in Bank 1.

For those who have the ALS CP/M Card, revision B, the "B" and "C" BIOS revisions all seem to put the start of common memory at 6000H.

The points I raise are relatively minor. I think readers who may have the ALS CP/M Card and have wondered how to do interbank accesses may be interested in the APINIT routine.

My appreciation to Albert Woodhull and to DDJ for the "BANKSWAP"

article.

Jerry Levy
1129 Dundee Dr.
Dresher, PA 19025

Help!

Dear DDJ,
I have been a subscriber to *Dr. Dobb's Journal* for quite a while and find it a very useful tool for my Toshiba T100 CP/M system. However, I am having some difficulty finding software to run a Black Box model 1200H modem (300/1200 bps). Any information or direction I should pursue would be most appreciated. Thank you.

Edward Starke
27 Kerwick Ct.
North Wales, PA 19454

Donations?

Dear DDJ,
I represent a nonprofit, charitable organization that uses microcomputer equipment in virtually every aspect of its affairs. We would be grateful if your readers would consider contributing additional equipment. Donations of this sort can have substantial financial benefits. If you are in a position to contribute or would like more information, please write or call (collect if you like) (617) 495-9020.

Robert Epstein
Cambridge Center for
Behavioral Studies
11 Ware St.
Cambridge, MA 02138

Corrections

There were two errors in the diagrams for Alan D. Wilcox's article, "Bringing Up the 68000—A First Step" (DDJ, January 1986). In Figure 4 on page 66, the 68000 bus request (BR) should be pin 13. In Figure 5 on page 68, the lower trace should be REST.—ed.

APINIT:	LHLD	0001	;get bios pointer to warm boot
	LXI	D,20H*3	;offset 32 more jumps to APREAD jump
	DAD	D	;add them
	SHLD	APRD+1	;store APREAD jump address locally
	INX	H	;next jump is APWRITE
	INX	H	
	INX	H	
	SHLD	APWRT+1	;store that locally
	INX	H	;bump to CAPPLE bios jump (if implemented)
	INX	H	
	INX	H	
	SHLD	CAPPL+1	;store locally
	RET		
;			
APRD:	JMP	0000	;these are filled in by APINIT
APWRT:	JMP	0000	;other bank will RET
CAPPL:	JMP	0000	

Table 1

ORG 0100H

DDJ

New for 'C'

A "C" programmer's tool to increase screen development productivity for the IBM PC. Security checking and help screen display are available at both the screen and field level. The automatic conversion of data types, to and from ASCII screen format, and the many other productivity-oriented features, set ZVIEW apart from the rest.

Screen Painter Highlights:

- Border colors and all character attributes and colors are supported.
- Draw single or double lined boxes using preset key strokes.
- Two field sensitivity settings to facilitate the moving and adding of fields, without destroying existing field characteristics.
- Three types of fields are available: "Protected," "Unprotected" and "Heading." The number of fields is limited to 600!!
- Both 40 and 80 column screens are supported.

Optional Field Characteristics:

- Choose left or right justification, with zero or blank fill.
- Automatic key stroke conversion to upper or lower case.
- Edit fields to be numeric (signed or unsigned), decimal (zero to six decimals supported), alpha or alphanumeric.
- Display numerical values with or without commas inserted.
- All "C" data types are supported, including a special long value which is displayed as a decimal field.
- From and to range checking and character matching edit.
- Security level settings to restrict inquiring or updating of a field.
- Override ZVIEW's default tabbing sequence.
- Assignment of a single or multiple screen help file, to be displayed when the field level help key is pressed.
- Compare one field to three other fields on the current screen.

Program Interface Highlights:

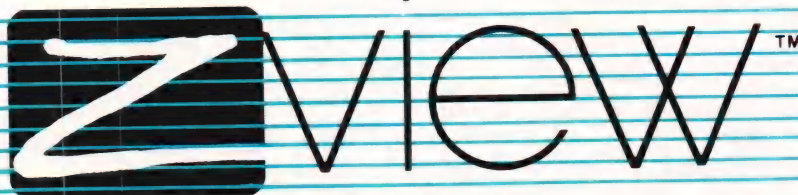
- Only seven run-time library functions control all aspects of the program to screen interface.
- Dynamically change any field characteristic at run-time.
- A wide range of run-time variables to further customize ZVIEW's operation.
- One call to ZVIEW's "Waitkey" function, performs all field edits and program to end user interface.
- Automatic data conversion of all data types to and from data structures and buffers. Data goes directly from data type to screen format and back, with one call each way.

Requirements:

- Microsoft 3.0 "C" or Lattice "C" compiler. (Call for additional information on compiler availability).
- IBM PC, XT, AT or compatible, MS/PC DOS, one 320k drive, a color graphics adapter and any 40 or 80 column display.

Price:

- \$245
Includes manual and a detailed program example.



TO ORDER CALL TOLL FREE 1-800-423-0930

Customer Service and Nevada residents:
call 1-702-798-5910



Or Write: Data Management Consultants
5325 So. Valley View Blvd. Suite #7
Las Vegas, NV 89118

Master Card, Visa or company
check accepted

IBM PC, XT, AT and PC-DOS are trademarks of International Business Machines.
MICROSOFT and MS-DOS are trademarks of Microsoft.
ZVIEW is trademark of Data Management Consultants

Circle no. 263 on reader service card.

DDJ ON LINE

This month I would like to describe in greater detail the purpose of the DDJ Forum on CompuServe, outline its structure, and give a brief description of the Forum Data Libraries.

The Forum can be reached by typing GO DDJFORUM from any CompuServe system ! prompt. The Forum, along with the Display Area (GO DDJ) serves as an electronic extension of the magazine. Our primary aim is to make available in electronic form most of the programming code published in the magazine. Other goals are to stimulate discussion between our readers and editors/authors/columnists and to serve as a general clearinghouse for information of interest to professional programmers.

Forum Architecture and Features

You enter the Forum at the Function Menu. From here you can access the Forum Message Boards, Conference Channels, Data Libraries, and Bulletins. In order to make use of these features, you should first learn a bit about the structure of the Forum.

The Forum is divided into seven Sub-Topics that roughly correspond to areas regularly covered in the magazine. Associated with every Sub-Topic are a Data Library, a Message Board, and a real-time Conference Channel. The Data Library (DL) is where files that come under the Sub-Topic are stored. The user may download files from the DL and may also upload files to a temporary holding area within the DL. These files are reviewed by the sysop and if found suitable are merged

into the DL itself. We invite your contributions. The Message Board allows messages relating to the Sub-Topic to be exchanged. The Conference Channel is where formal or informal conferences focusing on the Sub-Topic are staged.

Here's a concrete example. In the Forum, Sub-Topic 2 is called 16-Bit Toolbox. This Sub-Topic corresponds roughly to Ray Duncan's 16-Bit Software Toolbox column. Associated with this Sub-Topic is Data Library 2 (DL2), in which you can find the listings from Ray's column (along with listings from similar articles. If you want to leave a message related to Sub-Topic/DL2, you store that message under Sub-Topic 2. You can also read threads of messages under Sub-Topic 2 for discussion of issues raised by Ray's column. If we have a conference with Ray, it will be staged on the Conference Channel associated with Sub-Topic 2.

The subjects of the Sub-Topics/DLs are as follows:

0) General—General discussion, questions and help files.

1) C Chest—C programs from articles in *DDJ*; listings from Allen Holub's C Chest and Axel Schreiner's Unix Exchange columns.

2) 16-Bit Toolbox—8088/8086/80286 assembly language; PC/MS DOS and IBM PC related material; listings from Ray Duncan's 16-Bit Software Toolbox column.

3) 68K Toolbox—68000 assembly language; programs for the popular 68K micros (the Mac, Amiga, and ST).

4) CP/M Exchange—Z80 or 8080 assembly language; CP/M related material.

5) Pascal, Ada, M2—Programs written in the most popular structured languages (Pascal, Ada, and Modula-2).

6) Forth—Programs written in Forth.

The files in the Data Libraries are only program listings, that is, source code listed as ASCII text. We have decided to keep executable files in the DLs to a minimum. Our thinking is that it is primarily an educational resource and that the value of our code rests in the fact that people can read it and learn from it. Although executable files are useful, they do not teach the programmer anything.

Starting with the January 1986 issue, almost all listings from each new issue have been uploaded to the Forum. We also have been uploading code from back issues. The following is a partial list of these back issue listings along with other files contributed by Forum members. This list will have been substantially expanded by the time you read it.

XFR.C—A Christensen protocol file transfer program. Designed for 19.2K baud "nose-to-nose" transfers between computers, but also works very well in modem use at low speeds. XFR is compatible with most versions of Modem7. Written in Eco C for a Zorba CP/M machine, but can be easily modified. I/O primitives set up for I/O mapped Intel 8251A UART. By Donald Krantz. From #104 (June 1985).

BITMAP.C—A package of

general purpose bitmap management routines. Includes *makebitmap()*, *setbit()*, and *testbit()*. By Allen Holub. From C Chest, #104.

CROOT.C—A modified version of the Aztec CII (CP/M version) root module that allows that compiler to support pipes, redirection, quoted arguments, and command-line wildcard expansion. By Allen Holub. From C Chest, #101 (March 1985).

ECHO.C—A program that echoes its arguments to standard output. Useful for testing the *croot* modifications in CROOT.C and LOADER.ASM. By Allen Holub. From C Chest, #101 (March 1985).

GETARG.C—A general purpose command line parser. See also GETARG.H and STOIC. By Allen Holub. From C Chest, #103 (May 1985).

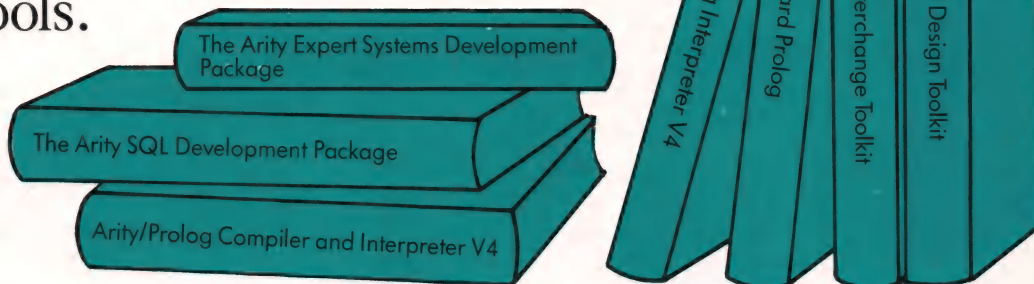
GETARG.H—A header file needed to use the routines in GETARG.C. By Allen Holub. From C Chest, #103 (May 1985).

LOADER.ASM—Assembly language support for CROOT.C. This routine allows you to chain to another program under CP/M (to do an *exec()* call). By Allen Holub. From C Chest, #101 (March 1985).

QSORT.C—A general purpose quicksort routine modeled after the Unix *qsort()* routine. See also SSORT.C. By Allen Holub. From C Chest, #102 (April 1985).

QUEUE.C—A package of general purpose queue management routines. Includes *makequeue()*, *del_queue()*, *enqueue()*, and *dequeue()*. By Allen

Why your next generation of products should use our 5th generation tools.



Arity's integrated family of programming tools allows you to combine software written in Arity/Prolog, the best of the fifth generation languages, with Arity SQL, the best of the fourth generation languages, and with conventional third generation languages such as C or assembly language to build your smarter application.

You can use Arity/Prolog to build expert systems using the Arity Expert Systems Development Package. Or to build natural language frontends. Or to build intelligent information management systems. Arity/Prolog lets you build advanced technology into your vertical applications package.

And more...

That's not the whole story. Arity's products are all designed to be fast, powerful, serious. Each of our products contains unexpected bonuses. Such as a one gigabyte virtual database integrated into Arity/Prolog. The most powerful of its kind on a PC.

Quality first. Then price.

In order to be the best, we had to prove it to our customers. Our tradition of quality software design is reflected in every product we sell. Quality first. Then price. And we always provide the best in customer support.

Our products are not copy protected. We do not charge royalties. We offer generous educational and quantity discounts. And we have a 30 day money back guarantee.

Try us to know that we keep our promise on commitment to quality and reliability. Try us by using our electronic bulletin board at 617-369-5622 or call us by telephone—you can reach us at 617-371-2422.

Or fill in this coupon. Whether you order today or not, let us send you full descriptions of our integrated family of Arity products.



We design and distribute high quality, serious application software for the IBM PC, XT, AT and all MS-DOS compatibles.

Please complete this form to place your order and/or request detailed information.

		Quantity	Info only
Arity/Prolog Compiler and Interpreter V4	\$795.00	_____	_____
Arity/Prolog Interpreter V4	\$350.00	_____	_____
Arity Standard Prolog	\$ 95.00	_____	_____
Arity SQL Development Package	\$295.00	_____	_____
Arity Expert System Development Package	\$295.00	_____	_____
Arity Screen Design Toolkit	\$ 49.95	_____	_____
Arity File Interchange Toolkit	\$ 49.95	_____	_____
TOTAL AMOUNT (MA residents add 5% sales tax) (These prices include shipping to all U.S. cities)		\$ _____	_____

NAME _____

SHIPPING ADDRESS _____

CITY/STATE/ZIP _____

TELEPHONE _____

Payment: ☐ Check ☐ PO ☐ AMEX ☐ VISA ☐ MC

Card # _____ Exp. date _____

Signature _____

ARITY CORPORATION • 358 BAKER AVENUE • CONCORD, MA 01742

Circle no. 121 on reader service card.



DDJ ON LINE

(Continued from page 14)

Holub. From C Chest, #104 (June 1985).

RK4.C—Fourth order Runge-Kutta integration of a single differential equation. This program along with RKTST1.C, RK4N.C, RKST1.C, and RKST2.C (also in the DLs) forms a system for RK4 integration of single or multiple differential equations. An excellent example of a scientific/engineering algorithm for numerical analysis implemented in C. By M. Roberts and A. Skjelum. From Toolbook of C.

SSORT.C—A version of *qsort()* that does a Shell Sort. This routine has been improved somewhat over the version

in K & R by using a gap size that isn't a power of two, as per Knuth. See also QSORT.C (in this same DL). By Allen Holub. See C Chest, #102 (April 1985).

STOIC—A string to integer conversion routine used by GETARG.C (in this same DL). Accepts hex, octal, and decimal representations and updates its argument to point past any parsed digits. By Allen Holub. From C Chest, (May 1985).

DUMPF.ASM—Filter to produce a formatted dump in Hex and ASCII. By Richard Markley. See also DUMP.C and DUMP.ASM. From #109 (November '85) 16-Bit Software Toolbox.

FSTCLN.ASM—Converts a word processor docu-

ment file into a standard ASCII text file. Similar to CLEAN.ASM and CLEAN.C but much faster. By Ray Duncan.

LJ.C—Utility to print a file on the Hewlett-Packard LaserJet. Prints pages "2-up" in Landscape Mode. Compatible with Microsoft C 3.0. By Joe Barnhart and Ray Duncan. Improved version of LJ.C from #107 (September '85) 16-Bit Software Toolbox.

ASCBIN.ASM—Subroutine to convert decimal or hexadecimal ASCII strings into their binary equivalents. By Ray Duncan.

BINASC.ASM—Subroutine to convert 32-bit binary number into an ASCII decimal string. Includes a general purpose 32-bit divide routine. By Ray

Duncan. From #101 (March 1985) 16-Bit Software Toolbox.

BREAK.ASM—Control-Break interrupt handler for Microsoft C programs. By Ray Duncan. From #107 (September 1985) 16-Bit Software Toolbox.

BREAK.C—Demonstration of the use of the Control-Break handler in BREAK.ASM. By Ray Duncan. From #107 (September 1985) 16-Bit Software Toolbox.

CLEAN.ASM—Filter to transform a word processor document file into a normal ASCII text file. Assembly language version. From #108 (October 1984) 16-Bit Software Toolbox. By Ray Duncan.

DDJ

Parlez-vous Pascal?

Mais oui.

After you've digested this, you will.

Introducing "Exploring Pascal: A Compiler For Beginners."

The fastest way on the planet to learn and use Pascal.

So you can put some of your programming ideas to work, to accomplish your specialized needs. And have the power of a pseudo-compiler, too.

It's fast because we make it as easy as possible.

Exploring Pascal is a unique, multimedia book and disk learning system.

The manual is very easy to read. Because you don't want to have to struggle with the

English language to learn a new computer language.

The software tutorials are interactive, and the exercises are computer graded. So you know how you're doing, every step of the way.

And there are animated demonstrations of important concepts.

400 screens of help in all.

To order, for the name of your nearest dealer, or for more information, just call the Ashton-Tate Publishing Group at 800-437-4329, Ext. 248.

If you're computer literate and are ready to step up to computer programming, it's really très bon.

Trademark/owner: Ashton-Tate. © 1986 Ashton-Tate. All rights reserved. Specifications subject to change without notice.



Circle no. 248 on reader service card.

Not Copy
Protected

DISCOVER THE LANGUAGE OF ARTIFICIAL INTELLIGENCE

PROLOG V

Interpreter for MS-DOS/PC-DOS

At last! A Prolog with enough muscle to handle real-world applications for UNDER \$100! Discover why Japan has chosen Prolog as the vehicle for their "Fifth Generation Machine" project to design intelligent computers.

CHOOSE FROM TWO GREAT VERSIONS:

PROLOG V-Plus

\$99⁹⁵

- ☐ More Than 100 Predefined Predicates
- ☐ Large Memory Model (to 640K)
- ☐ Floating Point Arithmetic
- ☐ 150-Page User's Manual and Tutorial plus Advanced Programming Documentation
- ☐ Co-Resident Program Editor
- ☐ Calls to Co-Resident Programs
- ☐ Text and Graphic Screen Manipulation

PROLOG V

\$69⁹⁵

- ☐ 70 Predefined Predicates
- ☐ Small Memory Model
- ☐ Integer Arithmetic
- ☐ 122-Page User's Manual and Tutorial

FREE FREE FREE FREE FREE FREE

Programming in Prolog
by Clocksin & Mellish, \$17.95 value
Available with purchase of PROLOG
V-Plus only. Offer valid through
March 31, 1986.

STANDARD FEATURES ON BOTH:

- ☐ Clocksin & Mellish-Standard Edinburgh Syntax.
- ☐ Extensive Interactive Debugging Facilities
- ☐ Dynamic Memory Management (garbage collection)
- ☐ Custom-Designed Binder and Slipcase

THE CHOICE OF UNIVERSITIES

Generous university site licenses and an excellent teaching tutorial and reference guide have made PROLOG V the choice of universities nationwide. Call for details.

Upgrade
to PROLOG V-Plus
for only the difference
in price plus a
handling charge.

NO RISK OFFER
Examine the documentation
at our risk for 30 days. If
not fully satisfied, return
with disk still sealed for
full refund.

PHONE ORDERS: 1-800-621-0852 EXT 468

☐ PAYMENT ENCLOSED \$ _____

CA residents add 6% sales tax

☐ CHARGE MY: ☐ MasterCard ☐ Visa

Card No. _____ Exp. Date _____

Signature _____

Mr./Mrs./Ms. _____

(please print full name)

Address _____

City/State/Zip _____

PROLOG V-Plus \$99.95
PROLOG V 69.95
UPGRADE ONLY 40.00
Return factory diskette and
\$30 plus \$10 Handling

SHIPPING:

\$ 5.00 U.S.
7.50 Canada
10.00 Caribbean,
Hawaii Air
20.00 Overseas Air

COD Orders Not Accepted
15 day check clearance



CHALCEDONY
SOFTWARE

5580 LA JOLLA BLVD.
SUITE 126 D
LA JOLLA, CA
92037
(619) 483-8513

Redirection—The /dev Directory, SWITCHAR, and Touch

I/O Redirection and /dev

I'd been avoiding adding redirection to the shell ever since I first wrote it. I thought it was just too hard to do. It turns out that I was wrong. So, I'm starting out this month by talking about how redirection works. I've already added this code to the version of the shell that *DDJ* is distributing.

Let's start with a little background. In Unix and in DOS, there's no difference between a file and a device from the point of view of the I/O routines. That is, all the DOS function calls that can talk to a file can also communicate with any DOS device, including the console and the printer. Unix supports a special directory called */dev* for this purpose. All devices are treated as files in the */dev* directory—for example, you can send output directly to a terminal by writing to that terminal as if it were a file. Each terminal has a unique name associated with it, such as */dev/tty01*.

To my pleasant surprise I discovered quite by accident that DOS supports this same mechanism, though it uses its own device names. (I wish they'd document some of this stuff.) For example, you can tell DOS:

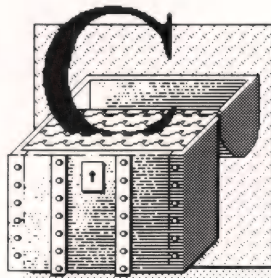
```
A> type foo > \devprn\
```

and it will send the output to the printer. Try it. The mechanism is supported even at the programming level. You can go ahead and open the

by Allen Holub

printer for output, and then write to it, by saying:

```
FILE *lpr;
lpr = fopen("/dev/prn", "w");
fprintf(lpr,
    "Quo usque tandem abutere");
fprintf(lpr, "... patientia nostra.\n");
```



All the normal DOS devices are supported—*/dev/con*, */dev/com1*, and so on. Your C I/O library actually doesn't know that it's writing to a device—it thinks it's accessing a file.

The console I/O functions take advantage of this mechanism. All the low-level I/O functions use a set of "file handles" that are maintained by DOS. A file handle is an integer value that's returned from DOS when a file or device is opened. This handle is then passed back to DOS every time you want to communicate with the associated file or device. In fact, three file handles (0, 1, and 2) are opened for you by DOS when it executes your program. Output sent to handles 1 and 2 goes to the screen, and input from handle 0 comes from the keyboard. The three I/O streams *stdin*, *stdout*, and *stderr* use handles 0, 1, and 2, respectively.

Handles 0, 1, and 2 aren't special. You can close them with a normal *close()* subroutine call. You can also reassign them to another file or device by opening them again. There are three ways to do this reassignment. First, when you close a file with *close()*, it frees up the associated handle. The next *open()* call will just grab the first available handle—if you close handle 0, the next *open()* call will use handle 0 for the newly opened file. For example, standard input can be reassigned with:

```
close(0);
if(open("Elizabeth", O_RDONLY) != 0)
    printf("Couldn't reassign
        standard input");
```

Here, standard input is closed. The

subsequent *open* call then uses the next available handle, which will be handle 0. You can make sure the reassignment worked as expected by testing the return value from *open* (the new handle) to make sure it's 0 (standard input). Though this method works if your I/O library is really Unix compatible, your compiler may not duplicate Unix exactly.

Another way of reassigning handles is to use the *dup2()* system call. The following example opens an existing file for appending and then changes standard error to write to that file rather than to the screen.

```
if( (fd = open("Bennet",
    O_WRONLY | O_APPEND)) == -1 )
    printf("Can't open Bennet\n");
else
    dup2( fd, 2 );
```

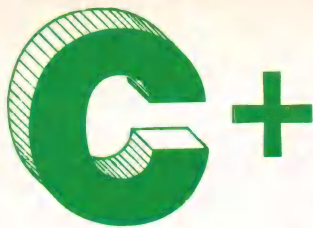
The file *Bennet* is opened for write, and then file handle 2 (standard error) is forced to reference *Bennet* rather than standard error (with the *dup2()* call). The file name can be any device, so if you had said */dev/prn* instead of *Bennet* then standard output would be sent to the printer instead of to the screen.

The third mechanism, and the easiest to use, is the *freopen()* system call:

```
freopen( filename, mode, stream )
char *filename, *mode;
FILE *stream;
```

The first two arguments are the same name and mode that you'd use with a normal *fopen()* call. *Freopen* will change the indicated stream, as well as the associated file handle, so that it now references the newly opened file instead of the original one. You can change *stderr* so that the error output is sent to the printer with the following:

```
if( !freopen( "/dev/prn", "w", stderr) )
    printf("Can't reopen stderr\n");
```

The Best C Book A Powerful C Compiler

One Great C Value \$39.95

A good C book just isn't complete without a good C compiler to go with it. That's why we give you both. You get a comprehensive 450 page book and a full feature standard K&R C compiler with the Unix V7 Extensions. The Book is loaded with examples that teach you how to program in C. And our fast one pass C compiler comes with an equally fast

linker so you don't waste a lot of time watching your disk drives spin. You also get a Unix compatible function library that contains more than 150 functions (C source code included). And if all that isn't enough, we offer you a 30 day money back guarantee. So what are you waiting for? The exciting world of C is just one free phone call away.

Language Features

- Data Types: char, short, int, unsigned, long, float, double
- Data Classes: auto, extern, static, register
- Typedef, Struct, Union, Bit Fields, Enumerations
- Structure Assignment, Passing/Returning Structures

abs
asm
asmx
atan
atoi
atol
bdos
bdosx
bios
biosx
calloc
ceil
cfree
chain
character
chdir
chmod
clearerr
close
clrscrn
cmpstr
conbuf
conc
cos
cpyst
creat
curblk
curlin
curscol
cursrow
cursoff
curson
delete
drand
exec
execl
execv
exit
exitmsg
exp
fabs
fclose
fdopen
feof
ferror
fflush
fgetc
fgetl
fopen
fread
free
freopen
fscanf
fseek
ftell
fwrite
getc
getch
putc
getchar
getcseg
getdseg
getd
putd
getdate
gettime
geti
puti
getkey
getmode
setmode
gets
getw
heapsiz
hypot
index
inp
insert
iofilter
isalnum
isalpha

Functions

isascii
isctrl
isdigit
islower
isprint
ispunct
isspace
isupper
itoa
keypress
left
len
log
log10
longjmp
lseek
malloc
alloc
mathtrap
mid
mkdir
modf
movmem
open
outp
peek
perror
poke
poscurs
pow
printf
putc
putchar
puts
putw
rand
read
readatr
reach
writech
sqr
readdot
writedot
realloc
rename
replace
repmem
rewind
right
rindex
rmdir
scanf
setbuf
setbufsiz
setcolor
setdate
settime
setjmp
setmem
sin
sound
sprintf
sqrt
srand
sscanf
stacksiz
str
strcat
strdup
strcpy
strlen
strncat
strncpy
strsave
system
tolower
toupper
ungetc
ungetch
unlink
write
writechs
xmembeg
xmemend
xmemget
xmemput
xmovmem
_exit

MIX Editor \$29.95

When you're programming in a high level language you need a high powered editor. That's why we created a programmable full/split screen text processor. It lets you split the screen horizontally or vertically and edit two files at once. You can move text back and forth between two windows. You can also create your own macro commands from an assortment of over

100 predefined commands. The editor comes configured so that it works just like Wordstar but you can change it if you prefer a different keyboard layout. The editor is a great companion to our C compiler. Because they work so well together we want you to have both. To make sure you do, we're offering the editor for just \$15 when purchased with the C compiler.

ASM Utility \$10

The ASM utility disk allows you to link object files created by Microsoft's MASM or M80 assemblers. Lots of useful assembly language functions are included as examples.

ORDERS ONLY
1-800-523-9520
IN TEXAS
1-800-622-4070

Canadian Distributor
Saraguay Software: 416-923-1500

NOT COPY PROTECTED

Editor	\$ _____ (29.95)	<input type="checkbox"/> PC DOS/MSDOS (2.0 or later)	Name _____
C	\$ _____ (39.95)	<input type="checkbox"/> IBM PC Single Side	Street _____
C & Editor	\$ _____ (54.95)	<input type="checkbox"/> IBM PC Double Side	City _____
ASM Utility	\$ _____ (10.00)	<input type="checkbox"/> Tandy 2000	State _____
TX Residents	\$ _____ (6.125% sales tax)	<input type="checkbox"/> 8 Inch	Zip _____
Shipping	\$ _____ (see below)	<input type="checkbox"/> Other _____	Country _____
Total	\$ _____	<input type="checkbox"/> CPM 80 (2.2 or later)	Phone _____
<input type="checkbox"/> Check <input type="checkbox"/> Money Order		<input type="checkbox"/> 8 Inch	
<input type="checkbox"/> MC/Visa* _____ Exp _____		<input type="checkbox"/> Kaypro II	
Shipping Charges: (No charge for ASM Utility)		<input type="checkbox"/> Kaypro 4	
USA: \$5/Order		<input type="checkbox"/> Apple (Z80)	
Canada: \$10/Order		<input type="checkbox"/> Osborne I SD	
Overseas: \$10/Editor • \$20/C • \$30/C & Editor		<input type="checkbox"/> Osborne I DD	
		<input type="checkbox"/> Morrow MD II	
		<input type="checkbox"/> Other _____	

MIX
software

2116 E. Arapaho
Suite 363
Richardson, TX 75081
(214) 783-6001

Ask about our volume discounts.

COMPUTOUGH

"Anyone
who wants to win MegaWars
has to dominate
entire planetary systems.
And me."

COMPUFUN

"You Guessed It!?" It's just like a TV game show.
Answer questions—win prizes.
And I can play right here
in the living room!"

C CHEST

(Continued from page 18)

All this is well and good, but how does it apply to redirection? In DOS, when a program spawns a child process (that is, when one program executes a second program with an `exec()`, `fork()`, or `spawn()` system call), the child process (the second program) inherits all the file descriptors (file handles) of the parent process. That is, if you are writing to a file and you spawn a child process, that child can continue writing to the same file without reopening it. This was causing problems in batch file processing because it turns out that the child can also close files that belong to the parent as an unwanted side effect of a normal `exit()` call.

Anyway, for some reason it had never occurred to me that `stdin`, `stdout`, and `stderr` (or more accurately the file handles 0, 1, and 2) aren't special. If they've been modified (with any of the mechanisms that I've just described) to talk to a file rather than to the console, they'll still be referencing that new file when the child inherits them. If the child tries to write to standard output, the child will actually write to the file. So, if you redirect standard input, output, or error in the parent process (in this case the shell), the child process will have its equivalent I/O streams redirected to the same place.

To my amazement all the foregoing actually works correctly in DOS. The code to implement redirection is in Listing One, page 58. Note that I've supported the following five Unix redirection modes (DOS supports only the first three of these):

- < *file*—Input is taken from the file rather than from standard input.
- > *file*—Standard output is put into the file rather than being printed on the screen. The previous contents, if any, of the file are destroyed.
- >> *file*—Same as > except that output is appended to the end of the file if the file exists.
- >& *file*—Same as > except both standard output and standard error are redirected.
- >>& *file*—Same as >> except both standard output and standard error are redirected.

SWITCHAR for DOS 3.x

A few months back I mentioned the `SWITCHAR=-` function. Putting the above in your `config.sys` file would cause the `-` to be used for command line switches rather than `/`. The `/` could then be used as a directory separator. Unfortunately, this feature, undocumented in DOS Version 2.x, was removed entirely from Version 3.x.

To some extent, writing the shell was my solution to this problem, but for those of you who prefer `command.com`, Tony LiCausi writes: "Commuters between DOS and Unix, take heart. Function call `0x37` of `INT 0x21` still supports 'switch the switch character.' If the `AL` register is set to 0, the function will return the current switch character in `DL`. If the `AL` register is set to 1, the switch character will be set to the character found in `DL`.

"There have been, and still are, drawbacks to setting the switch character to something other than `/`. Not all programs support the new character. `Command.com` and all its internal routines do accept it, however, as do *Join*, *Subst*, and *Format*. On the other hand, *Backup* and *Restore* do not (so be sure the switch character is set to `/` when you run these). New versions of the linker and so on don't support the switch character mechanism. Many of these programs, however, will accept both `/` and `-` as switch characters (for example, the C compiler driver, `CL`, accepts both)."

Listing Two, page 59, is a program of Tony's that sets or examines the switch character, depending on whether a new character is specified on the command line:

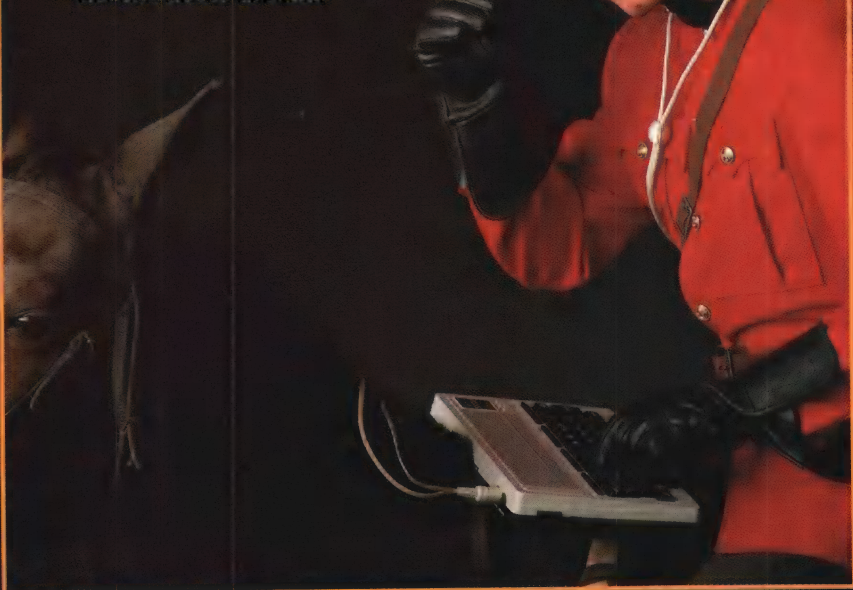
switch—Prints the current switch character.

switch c—Sets the switch character to `c`.

I moved Tony's program over to the Lattice compiler so that I could test it. If you're working with the `CC86` compiler, you'll want to use `sysint21()` rather than `intdos()`. Function `0x37` is still an undocumented function so you can expect it to disappear at the whim of Microsoft, but for now the program should be useful.

COMPU CRAZY

"Ready for an adventurous challenge?
We're a team. And Nellie
doesn't horse around."



COMPU SERVE GAMES

You never know
who you'll be up against
when you go online
with CompuServe.

To buy your CompuServe Subscription Kit,
see your nearest computer dealer.
Suggested retail price \$39.95.

To request our free brochure or order direct,
call or write: **800-848-8199**
(In Ohio, call 614-457-0802)

CompuServe®

5000 Arlington Centre Blvd.
Columbus, OH 43220

PROGRAMMING TOOLS

DEMO DISK
\$10.00
(credited towards
future purchase)



Version (4.0)

C-Plus (For IBM Microsoft C)

Turbo-Plus (For Turbo-Pascal)

Pascal-Plus (For IBM-Microsoft Pascal)

Compile 50 times faster.

Plus Software is a library of procedures crafted in assembly language and designed to enhance Turbo-Pascal™ and IBM-Microsoft Pascal. No royalties are required for applications developed using Plus Software. These procedures are extremely fast and combine automatically with your programs. The benefit to you is faster development and compilation time and smaller, faster application programs. Features include:

Multiple input/output field display map generation with edit masks

- Instant text write
- Instant attribute write (the fancy cursor maker)
- Display chunk retrieval
- Graphics text write (any size or position)
- Pop-up any time reference guide
- Pull Down Menu Maker
- Snow removal
- RamWindow
- Advanced keyboard control
- Multiple screen snapshots and restores
- File handle I/O
- Variable Length
- Sample programs with source code
- Printed manual

\$59.95 (Requires IBM PC and compatibles)



Screen Genie

The ultimate screen editor at any price.

- Creates and edits screens for programmers and non-programmers
- Resident pop-up facility for creating your own pop-ups
- Full typematic and cursor control. Insert, overwrite in two dimensions
- Paints/unpaints, draws/undraws in any direction. Easy to use selection menus
- Copy/move lines or window pieces with forgiving adjustment feature.
- Save and edit any size windows.
- SCREEN GRABBER lets you save and edit any screen from any application
- CREATES FIELDS AND GENERATES CODE FOR PROGRAMMERS
- Creates screens as executable files for non-programmers
- Creates load files (for any language), .obj files, .com external procedures
- Non-copy protected

\$69.95 (Requires IBM PC and Compatibles)



Turbo-Spawn

(For Turbo-Pascal)

\$39.95

(Requires MS PC DOS)

Turbo Spawn lets you execute any size program including another Turbo Pascal program or Dos commands without leaving current program and continue with next instruction. **Breaks the 64K barrier.**

Nostradamus Inc.

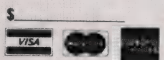
Software
sourcenry

Please send me:

- ☐ C-Plus \$59.95 ☐ Check ☐ Money Order Name _____
- ☐ Turbo-Plus \$59.95 ☐ Visa/MasterCard/Amex _____
- ☐ Pascal-Plus \$59.95 No. _____ Address _____
- ☐ Screen Genie \$69.95 _____
- ☐ Turbo-Spawn \$39.95 _____
- ☐ Demo Disk \$10.00 Exp. Date _____ City/State/Zip _____

Total amount

U.S. shipping and handling included
Outside U.S. pays postage



Nostradamus Inc. / 5320 South 900 East, Suite 110
SLC, Utah 84117 / Order by phone (801) 261-0769

Circle no. 251 on reader service card.

C CHEST

(Continued from page 21)

A Bug in mk and a Touch Utility

A make utility (called *mk*) was printed in this column in August 1985. Allen Orcutt writes: "I discovered that *mk* made an unnecessary null system call as the last action of every make. The problem is the *for* loop on line 364, which terminates on finding a null pointer at **linev*. I changed it to terminate on finding a null string at ***linev*.

```
for( linev =
    snode do->this; **linev; linev++ )
```

is the new line 364."

Michael Yam sent in a useful adjunct to the make utility, a version of the Unix utility *touch* (Listing Three, page 59). *Touch* changes the last-modified date and time fields to the current date and time. It's useful for forcing make to behave in certain ways. For example, if you change nothing but comments in a group of .c files, you can touch all the .obj files to stop make from remaking the entire program. If you touch a nonexistent file, the file is created with zero length.

Michael writes: "Add a nice touch to your make utility. Note that instead of calling DOS functions to actually modify a file's date and time stamp, I indirectly let MS DOS do the job by opening, modifying, and closing the file."

The "modification" is really a NOP. Michael reads the first byte from the file and then writes it back to the same place.

Availability

The redirection routines are part of the shell that *DDJ* is currently shipping (see advertisement, page 122). All other code is available on CompuServe (type: go *DDJ*).

DDJ

(Listings begin on page 58.)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 1.

THINK

FAST!

**If you're a C programmer
you could be a more productive C programmer.**

**Introducing Lightspeed C™ for the Macintosh™
from THINK Technologies, Inc.**

Lightspeed C is a compiled programming environment for the Macintosh™ that gives you speed, convenience, and top quality code generation, too.

With Lightspeed C, turnaround is 1000% faster. Time to build from scratch is 3 times faster. Time to link a typical 15,000 line program is 5 seconds! And generated code quality is better than any on the market.

Best of all, Lightspeed C's, integrated Edit-AutoMake-Launch environment makes turnaround a one-step process.

If you want to produce higher quality results with less time and effort, send for Lightspeed C today.

The above statements are based upon benchmarks for creating an executable version of XLISP v 1.4 (16.5K source lines) from scratch and by modifying, re-compiling, and re-linking one source file. Comparisons were performed using a 512K Macintosh with a 10MB Hyperdrive.*

Generated code size (in bytes)
Program build time (in secs.)

a. compile
b. link-to-run
c. TOTAL pgm build
Turnaround time (in secs.)
(time to make a change to module xlcont.c)

Consular (MacC V4.0)	Aztec (V1.06G)	Megamax (V2.1)	Lightspeed (V0.40)
36770	34566	37698	33870
887	654	354	194
99	49	95	5
986	703	449	199
159	108	127	9

☐ Send me Lightspeed C™
fast. \$175.00 for each
non-copy protected compiler.

☐ I need more to think
about, send me information
about Lightspeed C.™

Mail to:

THINK Technologies
420 Bedford Street
Lexington, MA 02173
Or call 617-863-5593

NAME

TITLE

COMPANY

ADDRESS

CITY

STATE

ZIP

TELEPHONE

☐ CHECK ENCLOSED
☐ MC ☐ VISA ☐ AMEX ACCT. #

EXP.
DATE

SIGNATURE

BRIE

The Boca Raton Inference Engine

by Robert Jay Brown III

A rule of inference is a procedure that, when applied to a set of hypotheses, produces a conclusion that logically follows.

PROLOG has been gaining in popularity recently partly because of the Japanese fifth generation project, in which PROLOG has been chosen as the primary development language. It's quite different from LISP, the other language frequently used in artificial intelligence research, because it is knowledge-oriented rather than procedure-oriented. LISP and PROLOG each have certain advantages for knowledge processing applications. In this article, I introduce you to PROLOG programming, LISP programming, and the underlying mechanisms involved in a PROLOG interpreter. A working micro-PROLOG program demonstrates how logic programming can be used for database applications. (Micro-PROLOG is a product of Logic Programming Associates, London.) A working muLISP program is used to explain the factoring, resolution, and paramodulation rules of inference. (MuLISP is a trademark of The Soft Warehouse, Honolulu.) I also describe a refinement of the resolution algorithm that could form the basis for a viable PROLOG interpreter. I explain the PROLOG deduction cycle and backtracking and the concept of a programming environment.

What Are Rules of Inference?

In the process of proving a thing to be correct or incorrect, we normally use what is popularly called logic. In everyday parlance, the argument used to prove a point is not always a valid argument: sometimes incorrect reasoning is employed. If the reasoning is incorrect, then the conclusion could be incorrect. It is the goal of mathematical proofs and computer programs to produce correct results.

Aristotle was the first person to attempt a formalization of logic, giving a set of 19 rules that could be used to

formulate a correct, or proper, argument. These rules are called syllogisms. The following example is taken from A. Bundy's *The Computer Modeling of Mathematical*

*Reasoning.*¹

Consider the argument:

All Ancient Greeks were perfect.

Aristotle was an Ancient Greek.

Therefore, Aristotle was perfect.

This is an instance of the argument:

All Ps were Q.

X was a P.

Therefore, X was Q.

The above syllogism using P, Q, and X is the *Darii* syllogism. When we substitute "Ancient Greeks" for the variable P, "perfect" for the variable Q, and "Aristotle" for the variable X, we arrive at the first argument.

It is important to note that the validity of the syllogism alone is insufficient to guarantee the truth of the conclusion: the hypotheses must also be true. A syllogism, such as one of the 19 Aristotelian syllogisms, is said to be a rule of inference. In this article I concentrate on the rules of inference rather than on the assumptions about which the inferences will be made.

Aristotle's 19 syllogisms do not exhaust all possible argument forms, although until George Boole invented propositional logic in the nineteenth century it was believed that they did. Propositional logic allows for the construction of complex hypotheses by using the connectives AND, OR, and NOT. But propositional logic does not allow for the kinds of substitution that Aristotelian logic permits. It took Gottlieb Frege to invent predicate logic, which combines the best of both worlds.

Robert Jay Brown III, 5225 NW 27th Ct., Margate, FL 33063

Thus, when we take the two statements "All Ancient Greeks were perfect" and "Aristotle was an Ancient Greek," we see that they can fit the form of the Darii syllogism if we set $P = \text{"Ancient Greeks,"}$ $Q = \text{"perfect,"}$ and $X = \text{"Aristotle."}$ The conclusion, "Therefore, Aristotle was perfect" follows by substitution into "Therefore, X was Q ." Thus we can arrive at a working definition of a rule of inference: a rule of inference is a procedure that, when applied to a set of hypotheses, produces a conclusion that logically follows from them.

The process of finding substitutions like the above is called unification, and the unification algorithm is at the heart of all mechanized rules of inference. Given an algorithm for unification, the implementation of a mechanized rule of inference is mainly a problem of control.

The implementation of an inference engine is the automatic application of a set of inference rules to prove a theorem. The purpose of such an inference engine may not be readily apparent, but because it will find substitutions for the variables, finding a proof can find the substitutions that make a statement true, and these substitutions can be the real answers we are looking for.

Predicate Logic

A logic programming system is the conventional way to use an inference engine. The PROLOG language is another way. (See page 36.) In both cases, the input to the programming system is expressed in clauses. In the case of PROLOG, the clauses are of a special form: Horn clauses, which are clauses in disjunctive normal form that have only one positive literal. They are the basic statements of PROLOG.

The example on page 36 and Listing One (page 62) uses the Simple syntax option for micro-PROLOG. Listing Three (page 66) shows the same family tree example as Listing One (the order is different, but the clauses are the same), but it is in the standard micro-PROLOG syntax, which is very much the same as the syntax of LISP. (See page 37.) Each predicate is represented as a set of LISP-like lists, one for each Horn clause. The form for a predicate reference (called an atom) is:

```
( <predicate-name> <parameter-1>
... <parameter-n> )
```

where each of the parameters are constants, variables, or Skolem functions. Skolem functions are devices used to express the idea of an existentially quantified variable. This is a case in which we are trying to express the idea that there is at least one instance that satisfies the clause. Normally in logic programming or PROLOG, the variables are universally quantified, which means that the clause must be true for all possible instances of the variable.

The form of a Horn clause is:

```
((<head-literal>) (<literal-1>) ... (<literal-n>))
```

where each of the literals is a predicate reference. The head-literal is the part that was to the left of the *if* in the simple syntax, and the other literals were to the right, joined by *and*.

The clause:

x child-of y if y father-of x

would be written in the normal syntax as:

```
((child-of  $x$   $y$ ) ( $\leftarrow$  father-of  $y$   $x$ ))
```

Some other PROLOGs (such as Waterloo PROLOG)² use a syntax like:

```
child-of( $x,y$ )  $\leftarrow$  father-of( $y,x$ )
```

where the \leftarrow is supposed to look like the arrow that mathematicians use for implication. It is pronounced "if," just like the simple syntax version.

Standard first-order predicate calculus uses several different forms, but the one I use here is disjunctive normal form. In this form of predicate logic, all the predicates in a clause are connected by the OR operator \vee , and all the clauses are implicitly ANDed together. Each predicate together with its arguments is an atom, and an atom together with an optional logical negation, or NOT operator \neg , is called a literal. Writing the above example in disjunctive normal form, we get:

```
child-of( $x,y$ )  $\vee$   $\neg$ father-of( $y,x$ )
```

This clause must be true, so if it is true that y is the father of x , then $\text{father-of}(y,x)$ is true, making $\neg\text{father-of}(y,x)$ false. The only way that the clause can be true if all its literals except one are false is for the remaining one to be true. Thus, $\text{child-of}(x,y)$ is forced to be true.

In clausal form logic, the literals of a clause are ORed together and the clauses are ANDed together. The result must be true for the clauses to be consistent. If we take a set of clauses known to be consistent and add one more clause and the inference engine finds the resultant set of clauses to be inconsistent, then we have refuted that last clause. Thus, to prove something, we add its denial to the set of clauses and turn the inference engine loose on it. If the inference engine finds the augmented clause set to be inconsistent, then we have refuted the denial of the test clause, thereby proving it.

In micro-PROLOG, the first literal, or head-literal, is the only positive literal in the clause, and the remaining literals all have negative signs. In generalized clause form logic, clauses can have all positive literals, all negative literals, or any mix of the two. Because $(x/y) = (y/x)$, order is not important to the logic. Likewise, because $a \& b = b \& a$, the ordering of clauses is not important either. In PROLOG, however, the ordering of the clauses can have a profound effect on the execution time of a problem and in some cases can cause the program to fail to terminate altogether. Furthermore, some of the nonlogical, procedural aspects of practical PROLOGs force an explicit dependence on clause order.

The Unification Algorithm

The process of finding a substitution that will make two clauses the same is called unification. The result of a unification is a substitution or binding environment such that when each clause is realized in that environment, the two clauses will be the same. I stated earlier that uni-

fication is at the heart of all of the inference rules. I will now describe how unification works.

A binding environment is a list of pairs, where the first element in each pair is changed to the second element in its pair wherever it occurs in the two expressions being unified. For example, take the substitution, or binding environment *change x into y*, and realize the expression $f(x) / \sim g(x,y)$ in this environment. This gives the expression $f(y) / \sim g(y,y)$ as the result. As a more difficult example, take the substitution *change x into f(y,z) and change z into p(a,g(b))*, then realize the expression $g(x) / q(x,y,z)$ in this environment. This gives the expression:

$g(f(y,z)) / q(f(y,z),y,p(a,g(b)))$

which further reduces, by applying the same substitution again, to:

$g(f(y,p(a,g(b)))) / q(f(y,p(a,g(b))),y,p(a,g(b)))$

The substitution is performed repeatedly until no further substitution can take place. This is sometimes called the recursive realization of an expression under an environment. You can see that there are no real limits on the complexity of the substitutions that can take place.

One situation that must be prohibited is the case where a subexpression of the source term in a substitution occurs in the substituted term. For instance, consider the substitution *change x into f(x)* when applied to the expression x . After the first application of the substitution, we get $f(x)$, but we can apply the substitution again to produce $f(f(x))$ and again to produce $f(f(f(x)))$. This process will continue to produce a sequence of $f(f(f(\dots f(x))))$ that goes on forever. When this situation arises, the realization process fails to terminate; therefore, we prohibit the application of a substitution of this sort. This *occurs* situation is not always so easy to detect. For instance, the substitution:

change x to y and change y to z and change z to x

is likewise prohibited. The offending substitution may be deeply nested in the expressions being substituted into, to, and from.

The problem of unification is to find a substitution that makes two expressions the same when they are realized under that substitution. We can only unify expressions by substituting values for variables; we cannot substitute to change the value of something that is not a variable. For now, we will use the letters x , y , and z to stand for variables. For instance, the expressions $f(x)$ and $f(a)$ may be unified by the substitution *change x to a*, where x is a variable and a is a constant. It is equally valid to substitute one variable for another, as $f(x)$ and $f(y)$ may be unified by the substitution *change x to y*.

Things can certainly get more complicated. Unify:

$f(x,y) / g(a,z)$ and $g(a,b) / f(z,c)$

We can do this with the substitution:

change x to z, y to c, and z to b

Of course, not every expression can be unified with every other expression. If the numbers of terms in the expression do not match, no substitution in the world can unify them. For instance, it is not possible to unify the expressions $f(x)$ and $f(y) / f(z)$, even though the substitution *change x to y and change y to z* will make them algebraically equal because unification does not know the rule $(x / x) = x$. Likewise, if the predicate names do not match, unification is impossible, as in $f(x)$ and $g(x)$. A constant cannot be unified with anything except a variable or itself, making the unification of $f(a)$ and $f(b)$ impossible; however, a variable can be unified with an arbitrarily complex expression, as in:

$f(x)$ and $f(g(a,p(z),y))$

which results in the substitution:

change x to g(a,p(z),y)

Unification must produce the binding environment that defines the most general substitution, or unifier; when the two input expressions are realized under it, the unifier will unify these two expressions, making them the same expression. The most general unifier, or MGU, will not make a substitution unless it needs to. For example, unifying $f(x,y)$ and $f(x,7)$ should produce the substitution *change y to 7*. The two expressions could be unified with the substitution:

change y to 7 and change x to FOO

but that would not be as general as leaving x as a variable, because the expressions will unify without substituting any value for x .

In the following LISP implementation of unification, I represent a variable as any LISP-atom that begins with an asterisk. Thus:

`*x, *X, *THIS_IS_A_LONG_VARIABLE_NAME`

and even `*` alone constitute valid variables.

A binding environment is represented by a list of dotted pairs. The substitution:

change x to y and change z to f(x)

is represented by the list `((x.y) (z.(f x)))`, which will be printed by the LISP interpreter as `((x.y) (z f x))`.

A clause is likewise represented by a list. The clause:

$f(x) / \sim g(y,p(a,z)) / q(b,z)$

is represented by the list:

`((f x) (~ (g (y (p a z))) (q b z)))`

Listing Four (page 66) defines a set of muLISP functions,

Now dBASE is bilingual.

Announcing a second language
for dBASE.[®]

C.

Now you can add richer, faster
features to the dBASE you know and
love with "dBASE Tools for C."[™]

So you can continue to program
in the dBASE programming lan-
guage, and yet have state-of-the-art
calc speed and unique fast-painting
graphics.

Here's your tool kit:

A basic engine that links C,
special C libraries, and your own
C functions to dBASE applications.
(It supports Lattice[®] C, Microsoft[®] C,
and Manx Aztec[™] C.)

Arrays management and a C
library of financial, mathematical,
and statistical functions come with
the Programmer's Library.

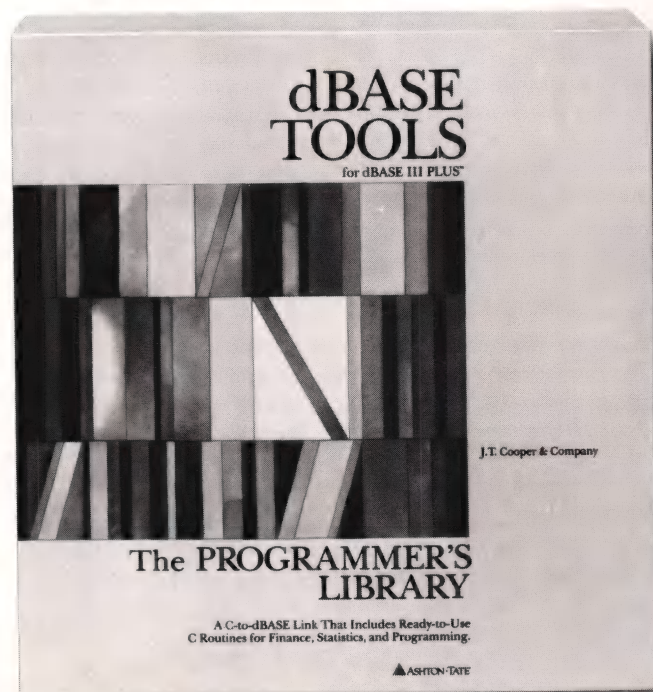
And the Graphics Library
includes interactive business graphics

like bar graphs, pie charts, exploded
pie charts, marked point graphs, line
charts, and XY charts.

To order, for the name of your
nearest dealer, or for more informa-
tion, call the Ashton-Tate Publishing
Group at 800-437-4329, Ext. 242.

Sure you need C to be on the
leading edge.

But you don't have to give up
dBASE to get it.



Trademarks/owner: Ashton-Tate, dBASE/Ashton-Tate. Microsoft/Microsoft, Inc.; Lattice/
Lattice, Inc.; Aztec/Manx Software Systems Inc.
© 1986 Ashton-Tate. All rights reserved. Specifications subject to change without notice.

Circle no. 242 on reader service card.

including *UNIFY*, a simple unification algorithm, according to J. A. Robinson.³ *UNIFY* calls *EQUATE*, which calls *UNIFY*, and so on, in a sort of ping-pong recursion. *ULT* finds the ultimate successor to a variable by first checking if that variable is defined in the environment. If it is, then its ultimate successor is the ultimate successor to its immediate successor, which is computed by *IMM*. *IMM* checks to see if the environment is *NIL*; if not, it checks if the variable is defined in the first binding pair in the environment. If the variable is not bound in the first pair on the environment, then *IMM* calls itself recursively to see if the variable is bound in any of the rest of the pairs in the environment.

EQUATE extends the environment by adding substitutions to it to make the two expressions the same. If the two expressions already are the same, then *EQUATE* returns the present environment. If the first expression is a variable and this variable does not occur in the second expression under the present environment, then the environment is extended by *CONS*ing a new substitution pair to the front of it. If the variable does occur in the second expression, as indicated by *OCCURS*, then the unification is impossible, and the value *IMPOSSIBLE* is returned instead of a binding environment.

The *occurs* check is not present in *PROLOG* because it eats up a lot of computer time, but strictly speaking it is necessary to prevent the interpreter from getting caught up in a recursive black hole from which it will never return. In the words of the "Adventure" program, "... you are likely to fall into a pit" if the *occurs* check is absent. In *PROLOG*, the programmer must ensure that an *occurs* situation never occurs. This is usually not too difficult in practice, but in order to prove that an inference rule works we need to know that the unification algorithm always works, and it only works all the time when it performs the *occurs* check. Without the check, the algorithm could fail to terminate, which would thereby violate one of the requirements of an algorithm.

The *RECREAL* (recursive realization) function instantiates the variables in an expression by performing the substitutions indicated by the environment. It does this by calling *ULT* for each variable in the expression, traversing the expression by recursive calls to itself, and building the resultant expression as it goes.

The *VARIABLEP* function is a predicate that returns *T* or *NIL*, depending on whether its argument is a variable. For our sample program in Listing Four, a variable is any atom whose print name starts with an asterisk (*). This is in accord with Waterloo *PROLOG*.⁴

In Listing Five (page 68) we see a sample run of *UNIFY* and *RECREAL*. The first example defines two clauses, *C1* and *C2*, and attempts to unify them by a call to *UNIFY*. *UNIFY* returns *IMPOSSIBLE* to indicate that the expressions cannot be unified. The recursive realizations of the two expressions are then the original expressions themselves. The second example defines a new *C2* and unifies it with the old *C1*, returning the environment $((x . a))$. This environment shows that the value *a* should be substituted for the variable *x* to unify the two expressions. Notice that

the recursive realization of the two input clauses is identical.

In the third example, *UNIFY* returns an environment consisting of $((x \text{ g } y))$, which is the same as $((x.(g \text{ } y)))$. This shows that the variable *x* must be instantiated with the value $(g \text{ } y)$ to make the two expressions the same. The fourth example shows a case in which the *occurs* check saves us. The fifth example shows a more complicated set of expressions. Note that *UNIFY* will work on arbitrarily complex expressions, limited only by the amount of memory available.

The Resolution Principle

Until recently, all formal rules of inference have been oriented toward human beings. An argument needed to be not only logically correct but also humanly comprehensible, because in a mathematical proof each step needed to be undisputable even though the conclusion might not be expected at all from the hypothesis. Such a proof might involve a great many deductive steps. When using a computer to deduce conclusions, one may be interested only in the results, not the details of each step along the way. In a 1965 paper describing a new rule of inference, J. A. Robinson called this new rule the resolution principle.⁵ Robinson's resolution principle is unique in that it is a complete rule of inference: no other rule is needed to reach any possible conclusion from a given set of facts. In actuality, an auxiliary rule called factoring is needed for certain cases, but resolution together with factoring forms a complete deductive apparatus.

The principle of resolution is really quite simple; the proof that it always works is difficult. See Robinson's aforementioned paper for a proof of the soundness and completeness of the resolution principle. Referring to Listing Four, we see the function *BINRES*. This function performs binary resolution on its arguments *CL1* and *CL2*, resolving on the literals at positions *N1* and *N2*, respectively. Recall that a literal is a predicate reference together with an optional negation sign. If this negation is present, we say that the sign of the literal is negative; otherwise, its sign is positive.

To resolve the two clauses *CL1* and *CL2* on their literals at positions *N1* and *N2*, perform the following.

1. Change the names of all the variables in one of the clauses so that the two clauses have no variables in common. This process is called taking variants.
2. If the signs of the two literals are the same, then the resolution cannot be performed, so return *IMPOSSIBLE*.
3. Otherwise, unify the two literals, without their signs. If the unification failed the resolution fails, so return *IMPOSSIBLE*.
4. Otherwise, delete the two literals from their respective clauses and return the OR of the resulting clauses. This is the resolvent of the clauses *CL1* and *CL2* about the literals *N1* and *N2*.

You can see that the resolvent contains two fewer clauses than the total between the two input clauses. If we can repeatedly cancel out literals until one of our resolvents is *NIL*, the empty clause, then we have found a contradiction. If we have added the denial of the thing we wish to

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library

routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.

- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.



SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- ☐ the C compiler for MVS software developers
- ☐ the C compiler for CMS software developers
- ☐ the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name _____
Title _____
Company _____
Address _____
City _____ State _____ ZIP _____
Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA.
27511-8000. Telephone (919) 467-8000, x 7000

4/86 DDJ

INFERENCE ENGINE

(Continued from page 28)

prove to a consistent knowledge base and we can resolve to produce the *NIL* clause, then we have proved what we set out to prove.

Consider the following two clauses:

$f(x,a) \mid g(x,y) \mid \sim f(y,b)$

and:

$p(x) \mid \sim g(a,x)$

We wish to form a resolvent between these two clauses. We can unify them about the second literal in each clause. Unifying:

$g(x,y)$ and $g(a,x)$

gives the MGU:

change x to a and change y to x

Now we can delete the two unified clauses from the union of the two original clauses to form:

$f(x,a) \mid \sim f(y,b) \mid p(x)$

But we are not finished yet; we must apply the substitution to the above clause to get the resolvent:

$f(a,a) \mid \sim f(a,b) \mid p(a)$

The Factoring Operation

I mentioned earlier that resolution was only complete if it included factoring. The following example is from R. Kowalski's *Logic for Problem Solving*,⁶ with the notation changed to fit our conventions:

$s(x) \mid s(y)$

$\sim s(u) \mid \sim s(v)$

The two clauses are inconsistent because they have instances:

$s(x) \mid s(x)$

$\sim s(u) \mid s(u)$

which, after removal of duplicate atoms, are directly contradictory:

$s(x)$

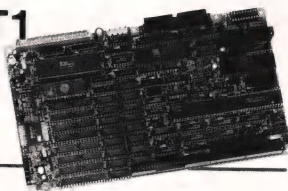
$\sim s(u)$

This is true because we can unify these last two clauses by the substitution:

change x to u

MSC-LAT1

\$649

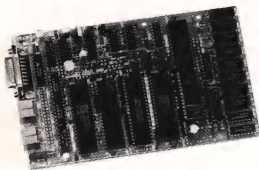


KAYPRO™

users can share the advantage to LAT1. Just take off your main KAYPRO board and put LAT1-K into your cabinet.

All advantage of LAT1 is yours now!

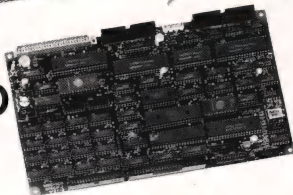
\$674



MSC-MTC

MSC-ICO

\$499



CP/M plus is a registered trademark of Digital Research Inc.
Z80 is a registered trademark of Zilog Inc.
Turbo Dos is a registered trademark of Software2000 Inc.
Mountain Side Computer and ZENET are trademarks of Southern Pacific Limited

Distributors

England-Quanta systems 01-253-8423
Denmark-Danbit 03-662020
Finland-BB Soft 90-692-6297
India-Betamatrix PVT Ltd. 0812-71989
Australia-LAMRON PTY. Ltd. 02-808-3666

Manufacturer and international distributor

SOUTHERN PACIFIC LIMITED

Sanwa Bldg., 2-16-20 Minamisaikai, Nishi, Yokohama, JAPAN 220
Phone: 045-314-9514 Telex: 3822320 SPACIF J

Advanced single board computer technology company

ZENET NETWORK through twist pair

- 6Mhz HD64B180 (Z80 upward compatible) 512K byte on board (256K installed, 384K RAM/DISK)
- LAN: ZENET port 800K baud CSMA CD twist pair bus type upto 500 meters HDLC
- Floppy: 3.5, 5 and 8 inch, d/s density, d/s sided and d/s track automatic density/format checking
- Hard disk: SCSI interface on board
- Video: 80 X 24 characters (color) and 640 X 200 pixels color graphic 128K byte video RAM character set is downloaded from disk
- Timer: battery back up calendar

- Serial: RS232C X 2 and TTL X 1
- Parallel: centronics type, 16 bit TTL, 7/8 bit keyboard port (32 characters FIFO)
- O.S.: Turbo Dos, MP/M (multiuser) banked CP/M plus (single user)
- Size: 10 X 6 inch 4 layered
- Assembled and tested
- BIOS source code available
- Completely faster than other Z80SBC

MSC-PCX

8088 expansion card for LAT1 soon available

WORLD SMALLEST COMPUTER

- Full personal CP/M system in palm 4mhz Z80 256K RAM (128K RAM/DISK)
- Serial: RS232C X 2 automatic baud rate checking
- Parallel: centronics type printer port
- Floppy: 3.5 inch micro floppy disk drive 800K byte (option 5, 3.5 inch drive d/s sided d/s track automatic density checking)

- O.S.: CP/M plus bank version
- BIOS source code available
- Completely faster than other Z80SBC

MSC-MTC/P

Full assembled pcb of MTC
Under \$189 in OEM quantity

\$299

Full featured CP/M plus system

- Z80 4mhz 128K Byte RAM Floppy: 3.5, 5 and 8 inch d/s density, d/s sided and d/s track upto 4 disk drives Automatic density/format check
- Serial: RS232C X 2
- Parallel: Centronics type, 16 bits I/O, 7/8 bit keyboard port
- Timer: battery back up calendar
- Video: 80 X 24 high speed CRT controller
- O.S.: CP/M plus bank version included
- Size: 10 X 6 inch 4 layered

- BIOS source code available
- DRI CP/M plus manual \$50
- New word word processor program for MSC-ICO ADD \$50
- Completely faster than other Z80SBC

MSC-HCS

Expansion card for ICO
RAM disk (upto 2M byte) and SCSI hard disk interface card for ICO with installation program

\$199

USA distributor

SOUTHERN PACIFIC COMPUTER PRODUCTS U.S.A., INC.

21 Altarinda Rd. Orinda, CA 94563
Phone: 415-253-1270

Dealer and distributor inquiries welcome

Circle no. 240 on reader service card.

to get:

$s(u)$

$\sim s(u)$

Because a statement cannot be both true and false at the same time, we have a contradiction.

However, no matter how many times resolution is applied to [the two original clauses] and their descendants, every resolvent contains exactly two atoms, and consequently no resolvent is the empty clause (which contains no atoms).

Factoring produces a new clause from an old clause by unifying two literals of like sign from that clause (without taking variants) and then performing the substitution and deleting one of the unified literals. See Listing Six (page 68), the first example, to see our factoring algorithm at work.

PROLOG does not need factoring because resolution is complete without factoring for sets of Horn clauses. A Horn clause is a clause with only one positive literal. In PROLOG this is the first literal, which is the consequent of the implication. Thus in the PROLOG clause x if y and z , x is the positive literal, and y and z are negative literals. If we rewrite the PROLOG clause in clause form logic, we have $x/\sim y/\sim z$. If y and z are both true, then $\sim y$ and $\sim z$ are both

false, so $\sim y/\sim z$ is also false. The only way the clause can be satisfied is for x to be true, which is the desired effect.

One theorem states that resolution is complete for Horn clauses without factoring. Another theorem is that any problem that can be expressed in clause form logic can be expressed in Horn clauses,⁷ and yet another theorem is that clause form logic (that is, first-order predicate calculus) is a complete basis for computation. PROLOG uses resolution on Horn clauses, which means that a PROLOG program is capable of computing anything that any other program is capable of computing.

The Inference Rule of Paramodulation

Even though resolution alone is sufficient to do anything we can do with any other programming language, it is not always efficient. One area in which efficiency often suffers is the processing of equality relations. An equality relation has a number of properties, such as symmetry, reflexivity, and transitivity. Each of these must be expressed as Horn clauses in order for a resolution inference engine to process equality relations. An inference rule known as paramodulation builds these properties of equality directly into the inference engine. See G. Robinson and L. Wos's discussion of paramodulation.⁸ The paramodulation refutation required 47 steps, and the pure resolution refutation required 136 steps. The process of paramodulation consists of the steps enumerated in the following list:

I Q C L I S P

THE CLOSEST THING TO COMMON LISP AVAILABLE FOR YOUR PC

RICH SET OF DATA TYPES

Bignums, for high precision arithmetic
8087 support, for fast floating point
Arrays, for multidimensional data
Streams, for device-independent i/o
Packages, for partitioning large systems
Characters, strings, bit-arrays

FULL SET OF CONTROL PRIMITIVES

flet, labels, macrolet, for local functions
if, when, unless, case, cond, for conditionals
Keyword parameters, for flexibility
Multiple-valued functions, for clarity
Flavors, for object-oriented programming
Stacks, for coroutines
Closures, for encapsulation

LARGE COMPLEMENT OF FUNCTIONS

Mappers, for functional programming
format, for output control
sort, for user-specified predicates
Transcendental floating point functions
String handling functions
Over 400 functions altogether

APPLICATION SUPPORT

Save and restore full environments
User-specified initializations
Assembly language interface

HARDWARE REQUIREMENTS

8088 or 8086 CPU, MSDOS Operating System
390K RAM or more

IQCLISP

5 1/4" diskettes
and manual \$300.00

Foreign orders add \$30.00 for airmail.
U.S. Shipping included for prepaid orders.

Integral Quality

P.O. Box 31970
Seattle, Washington 98103
(206) 527-2918

Washington State residents add sales tax.
VISA and MASTERCARD accepted.

EXTENDABILITY

defstruct, to add data types
Macros, to add control constructs
Read macros, to extend the input syntax
Extendable arithmetic system
Customizable window system

DEBUGGING SUPPORT

step, for single-stepping
trace, for monitoring
break, for probing
inspect, for exploring
Flexible error recovery
Customizable pretty-printer

MSDOS INTERFACE

Random access files
Hierarchical directory access
MSDOS calls

DOCUMENTATION

On-line documentation of functions
apropos
300-page indexed reference manual

1. Take two clauses. Call them *FROM* and *INTO*.
2. Take variants so that the clauses have no variables in common.
3. Search the *FROM* clause to find a positive equality relation.
4. Call one side of the equality relation *SEL*, the selection term, and call the other side *SUB*, the substitution term.
5. Unify *SEL* with a literal in the *INTO* clause. If this cannot be done, swap *SEL* with *SUB* and try this step again. If it still cannot be done, then unification fails, so paramodulation fails.
6. Instantiate both the *FROM* and *INTO* clauses with the substitution obtained from the above unification.
7. Replace the unified term in the *INTO* clause with the *SUB* term.
8. Delete the equality literal from the *FROM* clause.
9. OR the new *FROM* and *INTO* clauses together. This result is the paramodulant of the original two clauses.

If the equality relation cannot be found or if the unification cannot be performed, then no paramodulant exists. See Listing Four for a LISP implementation of paramodulation; see Listing Six for an example of its use.

The Boyer-Moore Structure-Sharing Resolution Algorithm

The resolution algorithm given above is operational and could be used as the basis for an experimental version of PROLOG, but it is terribly inefficient. J. A. Robinson described the resolution principle in 1965, but the first PROLOG interpreter did not appear until 1973. This was primarily because in 1972, R. S. Boyer and J. S. Moore published the first efficient implementation of a resolution algorithm.⁹ Their algorithm streamlines the process of taking variants and eliminates the copying of each clause every time it is resolved upon.

Whereas our earlier resolution algorithm used a variant-taking process that prefixed asterisks onto the name of a variable until it was unique, copying the clause as it did so, the Boyer-Moore algorithm uses a method known as indexing, which totally eliminates all the copying and uses simple arithmetic to form an index that is associated with each variable. In this implementation, variables start with a lowercase letter. Appending a prime is a conventional way of differentiating two variables of like name, such as *x* and *x'*. The Boyer-Moore index can be thought of as a count of the primes appended to a name to make it unique.

The advantage of the indexing system is that the new index needed to make a variable unique can be formed by a simple addition operation. Each variable is composed of a name and a corresponding index. The index is a positive integer. Instead of copying a variant clause while substituting the new variables, the structure-sharing algorithm extends the binding environment to keep track of the variants.

Listing Seven, (page 69) gives a muLISP implementation of the Boyer-Moore structure-sharing resolution algorithm. Listing 8 (page 70) gives a sample run of the algo-

rithm.

The algorithm uses a septuple, or data structure composed of seven elements. They are:

- | | |
|--------------|---|
| 1. LPAR: | The left parent. |
| 2. LLIT#: | The left literal number. |
| 3. RPAR: | The right parent. |
| 4. RLIT#: | The right literal number. |
| 5. NLITS: | The number of literals. |
| 6. MAXNDX: | The maximum index. |
| 7. BINDINGS: | The extension to the binding environment added at this level. |

This septuple represents a clause, either as a clause originally in the knowledge base or produced as a resolvent by the algorithm. The function *MAKECL* takes a clause as we would type at the keyboard and generates a septuple that is a list composed of the seven elements in order. The clause is pointed to by *LPAR*; *LLIT#* is zero; *RPAR* is *NIL*; *RLIT#* is zero; *NLITS* is the number of members (returned by the function *NMEMS*) in the clause; *MAXNDX* is one; and *BINDINGS* is *NIL*. This is the form of an input record. An input record is recognized by the predicate function *INRECP*.

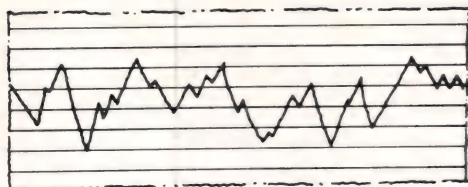
When we form a resolvent, we generate a new septuple, where *RPAR* and *LPAR* point to old septuples. *LLIT#* and *RLIT#* are the numbers of the literals resolved upon in the septuples pointed to by *RPAR* and *LPAR*. *NLITS* is the number of literals in this clause, which is two less than the sum of the *NLITS* cells in the septuples pointed to by *LPAR* and *RPAR*. *MAXNDX* is the sum of the *MAXNDX* cells in each of the parent clauses pointed to by *LPAR* and *RPAR*.

If we associate *MAXNDX* with each of the variables in the right parent of this new clause, we are guaranteed that all of its variables are unique as compared to the left parent. The function *GETLIT* retrieves a literal from a tree of these septuples and keeps track of the index and sign of the literal, returning them in the free variables *SIGN* and *INDEXG* along with the literal itself in the free variable *LITG*. (A free variable is a variable that is not local to the function in which it is used. It is somewhat analogous to a global variable in conventional programming languages.)

The Boyer-Moore structure-sharing unification algorithm works by returning *T* or *NIL* to indicate whether the unification was possible and extending the binding environment in the free variable *BNDEV*. *BNDEV* is the septuple in which the resolvent will eventually be stored. The algorithm is as follows.

1. If the two terms and the two indexes are equal, then return *T* without extending the environment.
2. If *TERM1* is a variable bound in the current environment, then substitute what it is bound to and try to unify the result with *TERM2*.
3. If the *occurs* check finds a bottomless pit, then return *NIL*.
4. Otherwise, force a unification by extending the environment and return *T*.
5. If *TERM2* is a variable, then swap *TERM1,INDEX1* with *TERM2,INDEX2* and try to unify that.

VEDIT[®] Plus Text Editor



The Navy charts new concepts with it... GM

engineers the future with



it...



facts

National Can preserves

with it... GE has bright

ideas with it... Here's why you
shouldn't be without it.



Every day, VEDIT PLUS helps thousands of programmers, writers and engineers get down to business.

So why do people who could have ANY text editor prefer ours? For a lot of reasons, including:

- **CAPACITY**—With VEDIT PLUS, file size is never a problem. And virtual disk buffering simplifies editing of even the largest files.
- **FLEXIBILITY**—VEDIT PLUS lets you edit up to 37 files simultaneously. So you can cut and paste. Edit programs. Edit text. Even perform numerous search/replace functions in several files without user intervention.*
- **CUSTOMIZATION**—With VEDIT PLUS, you can create your own on-line editing functions with keystroke macros. Develop your own on-line help screens. Determine and revise your own keyboard layout easily.

• **SPEED**—VEDIT PLUS not only works hard, it works fast. Faster, in fact, than any other text editor on the market.

• **EXPERIENCE**—Six years ago, CompuView revolutionized the concept of microcomputer text editing. And we've been improving our products and services ever since.

Special Offer: Order a VEDIT PLUS text editor for \$225 and we'll include our V-PRINT[™] document formatter—a \$120 value—absolutely free.

Call CompuView today at 313/996-1299. You'll be in good company.

CompuView[®]

CompuView[®] Products Inc., 1955 Pauline Boulevard—Suite 300, Ann Arbor, Michigan 48103, TELEX 701821

Available for PC DOS, MS-DOS, CP/M, CP/M-86.

*Free sort, compare, print and main menu macros included; optional 8080-8086 translator or mailmerge, \$50 each.

Circle no. 122 on reader service card.

INFERENCE ENGINE

(Continued from page 32)

6. Otherwise, try to unify the *CAR* of *TERM1,INDEX1* with the *CAR* of *TERM2,INDEX2*. If the unification fails, return *NIL*.

7. Otherwise, return the result of unifying the *CDRs* of *TERM1,INDEX1* and *TERM2,INDEX2*.

The comments in Listing Seven should make the code fairly easy to follow, although this is not a simple algorithm. The *occurs* check would be eliminated in a PROLOG implementation.

The Boyer-Moore algorithm could be the basis for a PROLOG implementation, but a viable PROLOG interpreter should not be written in a language such as LISP, which is interpretive itself. The structure-sharing resolution algorithm, rewritten in a portable, compiled, higher-level language such as C or Pascal, could form the basis for a practical inference engine in a Prolog i interpreter.

There is also an alternative to structure sharing: structure copying, also called non-structure sharing.¹⁰ Micro-PROLOG uses this method, which uses an indexing scheme for the taking of variants but does not share structure as does the Boyer-Moore algorithm.¹¹ Some claim that the structure-copying method can be superior to structure sharing on machines with a short word length, such as microcomputers. It remains to be seen which of these two methods is best for larger micros ad-

dress spaces of a megabyte and more.

Some PROLOGs, including micro-PROLOG, use special techniques to allow recursion to be as efficient as iteration, which is not available in PROLOG. These techniques are applicable in deterministic cases of recursion, such as was encountered in our first factorial algorithm, and are called tail recursion optimization. M. Bruynooghe discusses the implementation of these methods in *Logic Programming*.¹² Another related technique, success popping, is discussed in T. Bruynooghe's essay and in F. G. McCabe and K. L. Clark's *Micro-PROLOG 3.0 Programmer's Reference Manual*.¹³

To make a complete PROLOG system, however, requires a great deal more. First, resolution alone does not make an inference engine. A strategy is needed to determine what to resolve next. This strategy, called the search strategy, can be a breadth-first, depth-first, or heuristically guided search; the depth-first method is employed in PROLOG. The inference engine of PROLOG takes the query clause and resolves it with the first clause in the knowledge base that matches it in its head literal. Remember that the head literal is the only literal of positive sign. The query clause is the only clause with no positive literals. Therefore, PROLOG only needs to look at the head literals to try a resolution.

A practical PROLOG will have an efficient database retrieval system that allows it to quickly find candidate clauses that match a given literal with their head literal. Some PROLOGs even go so far as to index the knowledge

LISP

The preferred symbolic processing language of the Artificial Intelligence Community

catch the next micro-wave with

UO-LISP

Not "just another pretty dialect" but the most powerful implementation of LISP available in the micro market place. For the professional engineers, researchers, and educators, UO-LISP maintains the power and flexibility inherent in LISP while providing the expected functionality of mainframe LISP systems. (+) **UO-LISP steps beyond the competition and provides a real source to native code compiler.**

CPU Family	Operating System	Production System	Learn System	Production plus Learn System
8086	MS-DOS	150 ⁰⁰	85 ⁰⁰	185 ⁰⁰
	PC-DOS	150 ⁰⁰	85 ⁰⁰	185 ⁰⁰
	CPM/86	available soon	—	—
Z80	CPM	125 ⁰⁰	85 ⁰⁰	160 ⁰⁰
	TRSDOS	80 ⁰⁰	N/A	N/A

For MORE DETAIL AND TO ORDER:
Send for *FREE* brochures and order forms.

NORTHWEST COMPUTER ALGORITHMS

P.O. Box 1747, Novato, CA 94948
(415) 897-1302

base to the first two terms of the head literal, which greatly reduces the time to find a matching clause or determine that no match is possible. When a potential match is found, the two literals are unified, but the resolvent is not placed into the knowledge base yet. PROLOG then steps through the negative literals in the clause just resolved with, resolving each of them in turn with other clauses in the knowledge base. Each of these clauses is stepped through in similar fashion until the *NIL* resolvent is produced. Then execution continues at the previous level.

If a resolvent fails, PROLOG backtracks to the nearest choice point where it could have chosen a different Horn clause to resolve with and takes that branch of the search tree instead. Only after exploring the entire search tree and failing everywhere does PROLOG report failure. If PROLOG can deduce the *NIL* clause from the query clause and the knowledge base and if the knowledge base is consistent, the query clause is false. But the query clause is a negative clause, so the query condition is true.

When the *NIL* clause is deduced, the binding environment that produced it is available, and the substitutions provide additional information. In the family tree example, these substitutions are the answers we were looking for. The "which" predicate of the simple extension to micro-PROLOG continues after *NIL* is deduced to find all the answers rather than stopping on the first one.

Even after the search strategy and the database retrieval mechanisms are provided for, we would have a PROLOG interpreter, but we would not have a complete PROLOG system. A complete artificial intelligence program development system needs, as a bare minimum, a good interactive program editor and debugger, including a trace facility. These functions taken collectively form the programming environment. Also, built-in functions must be provided to perform arithmetic, input-output operations, list construction and decomposition, control, additions to and deletions from the knowledge base, and so on. The micro-PROLOG system provides all of these, and the programming environment is written in micro-PROLOG itself.

Acknowledgments

I would like to express my gracious appreciation for the guidance of Dr. Frederick Hoffman of Florida Atlantic University. My thanks also to Jim Romanowiz, also of FAU, for his help. Peter Steinfeld of Gould Computer Systems offered many helpful suggestions. Lastly, I wish to thank my patient wife Darlene, whose careful proof-reading corrected many of my linguistic atrocities. Nonetheless, any errors are my own.

Notes

1. A. Bundy, *The Computer Modelling of Mathematical Reasoning* (London: Academic Press, 1983).
2. G. M. Roberts, "An Implementation of PROLOG." Master's thesis, University of Waterloo, 1977.
3. J. A. Robinson, "Fundamentals of a Machine Oriented Deductive Logic," in *Introductory Readings in Expert Systems*, ed. D. Michie (New York: Gordon and Breach Science Publishers, 1982), 81–92.
4. Roberts, "An Implementation of PROLOG."

5. J. A. Robinson, "A Machine Oriented Logic Based on the Resolution Principle," *JACM* 12 (1965): 23–41.
6. R. Kowalski, *Logic for Problem Solving* (New York: Elsevier Science Publishing Co., 1979).
7. Ibid.
8. G. Robinson and L. Wos, "Paramodulation and Theorem Proving in First Theories with Equality," in vol. 4 of *Machine Intelligence*, ed. D. Michie (Edinburgh: Edinburgh University Press, 1969), 135–50.
9. R. S. Boyer and J. S. Moore, "The Sharing of Structure in Theorem Proving Programs," in vol. 7 of *Machine Intelligence*, ed. Meltzer and Michie (Edinburgh: Edinburgh University Press, 1972), 101–16.
10. C. S. Mellish, "An Alternative to Structure Sharing in the Implementation of PROLOG," in *Logic Programming*, ed. Clarke and Tarnlund (London: Academic Press, 1982), 99–106.
11. F. G. McCabe and K. L. Clark, *Micro-PROLOG 3.0 Programmer's Reference Manual*, CP/M version, 3d ed. (London: Logic Programming Associated, 1983).
12. M. Bruynooghe, "The Memory Management of PROLOG Implementations," in *Logic Programming*, ed. Clarke and Tarnlund (London: Academic Press, 1982), 83–98.
13. McCabe and Clark, *Micro-PROLOG 3.0 Programmer's Reference Manual*.

DDJ

(Sidebars follow; listings begin on page 62.)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 2.

ATTENTION

C-PROGRAMMERS

File System Utility Libraries

All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable.

BTree Library

75.00

- High speed random and sequential access.
- Multiple keys per data file with up to 16 million records per file.
- Duplicate keys, variable length data records.

ISAM Driver

40.00

- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use.

Make

59.00

- Patterned after the UNIX utility.
- Works for programs written in every language.
- Full macros, File name expansion and built in rules.

Full Documentation and Example Programs Included.

ALL THREE PRODUCTS FOR — **149.00**

For more information call or write:

softfocus

Credit cards accepted.

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

Dealer inquiries invited.

What Is an Inference Engine?

An Example Using micro-PROLOG

Consider the micro-PROLOG program given in Listing One. This is a simple relational (pun intended!) database system to keep track of all of my daughter's relations in the form of a family tree. Examination of the code reveals that the first section is a series of statements of the form *x father-of y*. This section is followed by a similar series of statements of the form *x mother-of y*.

Together these two sections constitute the knowledge base for the family tree. Statements of this form are called assertions or facts. Most of the *father-of* and *mother-of* relations are free of any variables; these are called ground clauses; however, a few of these relations are of the form *<statement-1> if <statement-2>* and typically contain variables. These are the conditional statements of PROLOG. The general form of a PROLOG statement, or clause, is:

<statement-1> if <statement-2> and ... <statement-n>

The procedures of PROLOG are called predicates, just like the procedures of FORTH are called words. A PROLOG clause is then more properly defined as:

<predicate-1> if <predicate-2> and ... <predicate-n>

The definition of a PROLOG predicate is either built into the interpreter or is defined by the applications logic programmer as a set of clauses, all having the same relation name, or predicate name. Thus, in the definition of *mother-of*, we find:

*lilian-givens mother-of x if
paul-sewall-jr father-of x*

This clause, together with all the ground assertions for *mother-of*, comprises the definition of the *mother-of* predicate. It is an interesting characteristic of logic programming in general, and PROLOG in particular, that the distinction between program and data becomes very fuzzy. In our family tree, were it to be written using a conventional database package, we would have to represent all *father-of* and *mother-of* relations as records in a father-of file and a mother-of file. These records would be data. If we wanted to write a procedure to handle the case of *lilian-givens*, we would probably discard the idea as impractical: the mother-of file contains pure data; if we needed to write a routine to compute something, the routine would be pure code. Never the twain shall meet! But in PROLOG, we can easily intermix data and program to do the job. We can use whichever approach seems most natural. In PROLOG, data are retrieved by unification; hence, we execute the data (or perhaps search the code) to find a match with the search pattern, and it matters not whether the data is pure data, pure code, or a mixture of both.

Looking at Listing Two (page 62), which shows a series

of query operations performed against the family tree, notice the form of the query:

which (<answer-pattern> <query-pattern>)

This is micro-PROLOG's query format. (Other interpreters differ in external details.) In the example, the answer pattern is always a single variable, but micro-PROLOG allows for arbitrarily complex answer patterns. For the first query, the inference engine attempts to find a value for *x* that will make the relation *x father-of robert-brown-iii* true. It finds when *x = robert-brown-jr*, the query pattern is true, so it displays this value as an answer. As it happens, I have only one father, so there are no more answers.

The next query illustrates another beautiful aspect of PROLOG: a procedure can be run in reverse, so to speak! The first example used a variable for the father, but this example uses a variable for the child. Again, because I have only one child, PROLOG displays my daughter's name and then says, "no more answers."

The next query uses an additional relation, *parent-of*, which is defined by the following two clauses:

*x parent-of y if
x father-of y
x parent-of y if
x mother-of y*

This says that *x* is a parent of *y* if *x* is either the father or the mother of *y*. We have defined a new relation in terms of two other relations. In a conventional database environment, this would either require creating a parent-of file or a *parent-of* subroutine, but in PROLOG we needn't distinguish between the two cases. When we ask PROLOG:

which (x x parent-of krystl-raquelle-brown)

PROLOG responds with:

Answer is robert-brown-iii
Answer is darlene-breedon
No (more) answers

In this case, there was more than one instance of *x* such that *x parent-of krystl-raquelle-brown* was true, so more than one answer was returned.

The *descendant-of* relation illustrates a simple case of recursion, where a predicate is defined in terms of itself:

*x descendant-of y if
x child-of y
x descendant-of y if
z child-of y and
x descendant-of z*

This relation will thread its way through the family tree and report all persons who are descendants of a given person. Notice how the first clause does not call *descendant-of* recursively. This clause provides the termination condition for the relation. If the first clause is not satisfied, the second clause will get us one step closer to

the case where it will be. Repeating via recursion will give us all the desired descendants. An *ancestor-of* relation is similarly defined in the listing.

Also in Listing Two are queries using the *aunt-or-uncle-of* and *cousin-of* relations that are defined in Listing One. Notice the simplicity of the definitions for these concepts, and compare the Prolog versions with the best you can do using a language such as BASIC, Pascal, or C. These languages are called procedural languages because they are used to describe a procedure to achieve a certain result as a sequence of steps such as: open files; read data; compute a lot; print answer. Prolog, on the other hand, is a nondeterministic language. It does not describe the sequence of steps needed to find an answer but merely describes the pattern that an answer must fit and lets the inference engine find one or more instantiations of the variables that satisfies the query pattern.

LISP, the Language of Artificial Intelligence

LISP is one of the oldest viable higher-level programming languages. It was revealed to the world by John McCarthy in 1960. It was at that time a totally new way to look at programming. Even today it is still very different from conventional programming languages such as BASIC, Pascal, C, FORTRAN, COBOL, or even assembler. Only other AI languages such as POPLER and PROLOG have similar properties. LISP is a functional language. A program consists of a set of function definitions, and the execution of a LISP program is the evaluation of a function. This is called the application of the function to its arguments, so LISP is called an applicative language.

In LISP notation, which has been called Cambridge Polish by some (distinct from the Reverse Polish of FORTH), the function $F(x,y,z)$ is written as $(F\ x\ y\ z)$.

LISP is an interpretive language, and the following dialog might occur between the interpreter and a programmer just starting to experiment with it (The \$ is the LISP prompt):

\$1
1

The programmer types 1. LISP responds that the evaluation of 1 is 1.

\$(PLUS 2 3)
5

Adding 2 and 3 to the function PLUS. LISP responds that the evaluation of the PLUS function with arguments of 2 and 3 is 5.

\$(TIMES 4 7)
28

Multiply 4 and 7. LISP evaluates the expression and returns 28.

\$(PLUS 4 (TIMES 5 9))

Micro to Mainframe Connection

The Model TC-50 1/2-inch tape subsystem provides a standard medium for transmission of mainframe data base information to PC users, while maintaining mainframe isolation and data integrity. Use ODI subsystems to import data to data base programs like dBase III.

The TC-50 subsystem also provides fast back-up capability as well as a device driver and interface software for popular compilers.

The TC-50 subsystem includes tape drive controller, cables and documentation. All ODI products carry a 30 day unconditional money-back guarantee, and are warranted for one year, parts and labor.

ODI

Overland Data, Inc.

5644 Kearny Mesa Road
San Diego, CA 92111
Tel. (619) 571-5555
Telex 754923 OVERLAND

Also Available —
XENIX tape
subsystems
for the
IBM AT



Circle no. 192 on reader service card.

Write-Hand Man

"Almost a Sidekick for CP/M"

Ted Silveira—Computer Currents, Aug. 27, 1985

"WHM is ingenious and works as intended"

Jerry Pournelle, BYTE Magazine, Sept. 1985 (c) McGraw-Hill

Now available for CP/M 2.2, CP/M 3.0 and ZRDOS!

The convenience of *Sidekick* on your CP/M machine! Trigger *Write-Hand-Man* with a single keystroke and a window pops open to run desk accessories. Exit *Write-Hand-Man* and both the screen and program are restored. Use with any CP/M program and most any CP/M machine. Takes only 5K of memory.

FEATURES

Notepad for quick notes
Appointment calendar
HEX calculator

File and Directory viewer
Quick access phonebook
14 digit decimal calculator

BONUS

Add applications written by you or others! No other *"Sidekick"* lets you add applications. Dump screens, setup printers, communicate with other computers, display the date and time. Let your imagination run wild!

\$49.95 (California residents add tax), shipping included. COD add \$2. Sorry, no credit cards or purchase orders. 30 day guarantee. Formats: 8 inch IBM, Northstar and most 5 inch (please specify).

Write-Hand-Man only works with CP/M 2.2, ZRDOS and CP/M 3.0 (please specify). Simple terminal configuration required. Not available for TurboDOS. Compatible with keyboard extenders, hard disks, and other accessories.

Poor Person Software

3721 Starr King Circle
Palo Alto, CA 94306
415-493-3735

Trademarks: *Write-Hand-Man* — Poor Person Software, CP/M—Digital Research, *Sidekick*—Borland International

Circle no. 169 on reader service card.

INFERENCE ENGINE

(Continued from page 37)

49

Getting a little braver, we try a nested expression, $4 + (5 * 9)$. LISP returns the expected result.

\$A
A

We see what the value of *A* is. *muLISP*'s auto-quoting gives an unassigned variable the value of its name.

\$(SETQ A 2)
2

SETQ is the assignment function. It returns the value of the second argument but has the side-effect of making the first argument take on that value also. LISP returns the value of 2.

\$A
2

Now we see what the value of *A* is. The *SETQ* worked.

\$(SETQ A (PLUS 3 4))
7

Perform $A = 3 + 4$. LISP returns a value of 7.

\$A
7

What is the value of *A*? Just as expected.

In LISP, the evaluation process can be inhibited by the use of the quote (') macro, which expands as follows:

'x expands as (QUOTE x)

The ' is just a shorthand because *QUOTE* is used so much. The *QUOTE* function inhibits evaluation. Let's try it out:

\$(SETQ A '(PLUS 3 4))
(PLUS 3 4)

Same as before but with the quote. LISP returns the quoted expression verbatim, with no evaluation.

\$A
(PLUS 3 4)

What is the value of *A*? It has the value of the quoted expression.

We can force evaluation by the *EVAL* function:

\$(EVAL A)
7

Evaluate the value of the variable *A*. The evaluation of (PLUS 3 4) is 7.

Notice how LISP does not care whether a variable has numeric or alphabetic information in it. In fact, the type of a variable can change during the course of execution of a program. LISP, and AI languages in general, are dynamically typed languages, which adds to both the power and beauty of LISP and a good many bugs in LISP programs. It is the responsibility of the programmer to be aware of what is in a variable when he or she uses it.

Until now, we have been talking about variables in a rather loose sense. In LISP, everything is either an atom or a list. An atom is a thing such as *A* in the above examples. Atoms have several characteristics that are maintained by the LISP system. An atom has a print name, which is the character string that we use to refer to the atom. The print name for *A* is *A*. An atom also has a value. The value of *A* above is changed several times by the use of *SETQ* and is displayed when we type the print name *A* at the interpreter prompt. The value of an atom may be another atom or a number, which is a special kind of atom, or a list. The *muLISP* interpreter assigns the print name of an atom to its value when the atom is created. To create an atom, you simply use it; it will be created automatically if it is not already present.

When we assigned (PLUS 3 4) to *A*, we assigned a list to *A*. This list is composed of the three atoms *PLUS*, 3, and 4. The last two atoms are numbers, the first one is not a number. The expression (PLUS 4 (TIMES 5 9)) is also a list. It has three members; they are the atoms *PLUS* and 4 and the list (TIMES 5 9), which is itself composed of the three atoms *TIMES*, 5, and 9. Thus we arrive at the following recursive definition for a list:

A list is an ordered sequence of zero or more atoms or lists.

This is fine, but what do we have when we have zero rather than more atoms or lists in our list? We have the empty list (). This list occurs so often that it has been given a name. The empty list is the value of the atom *NIL*.

Given a list such as:

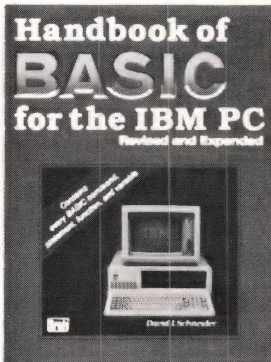
(A B (C D) (E (F G)) H (I))

we need the ability to take it apart and get at its components. The principal functions for decomposing a list are *CAR* and *CDR*. These rather cryptic names were once mnemonic in a machine-dependent way. *CAR* stands for contents of the address register, and *CDR* stands for contents of the decrement register. These were where the address pointers to the two halves of a list were stored on the machine that ran the original LISP. Needless to say, they have outlived their mnemonic significance, but history honors the architecture of the IBM 704 anyway.

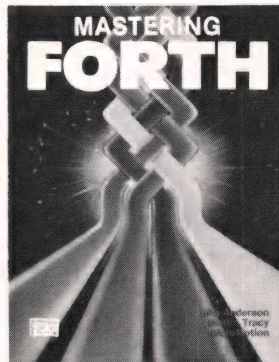
To extract the first component of a list we use the *CAR* function, and to extract the remainder of the list we use the *CDR* function. To make this easier to remember, notice that the *A* in *CAR* alphabetically precedes the *D* in *CDR*. *CAR* gets the first part; *CDR* gets the last part. Incidentally, *CAR* is pronounced just like the one you drive to work every day, and *CDR* is pronounced "could-er." Let's try some examples.

BRADY Speaks Your Language

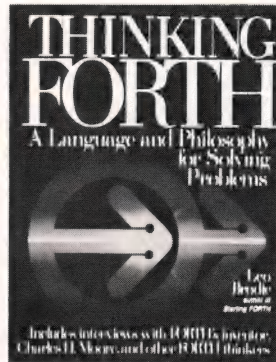
We can help you buff up your BASIC...Put a finish on your FORTH...
Conquer C...Succeed with Assembler...And more! Just call toll-free
or use the coupon to order today!



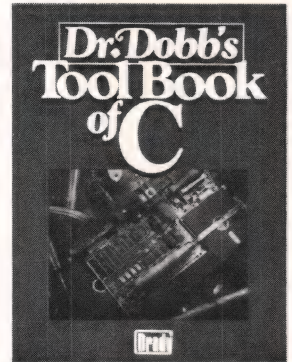
1. PC magazine calls it: "A truly remarkable book...A treasure trove of useful programming information for the IBM PC." \$22.95



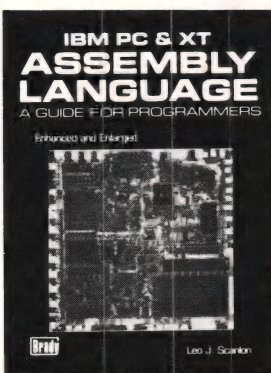
2. A step-by-step tutorial for the high-level, stack-oriented FORTH-83 standard. \$17.95



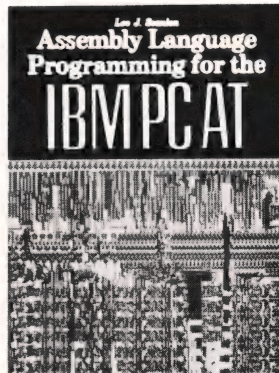
3. The FORTH authority, Leo Brodie, illustrates the elegant logic behind FORTH and shows how to apply specific problem-solving tools to software. \$16.95



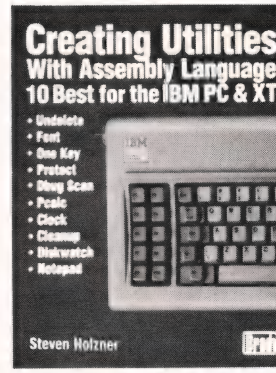
4. The best C articles from the highly respected Dr. Dobb's Journal dealing exclusively with C language and programming techniques. \$24.95



5. Our best-selling assembler book has been made even better! It now includes 30 assembler Macros and version 2.0 of the IBM Assembler. \$21.95



6. The author of our best-selling assembler books now demonstrates his detailed and accurate style on the 80286 chip. \$21.95



7. For the more advanced user familiar with Assembly language, here's an opportunity to unleash the power of 10 DOS-enhancing programs. \$21.95



8. Probes the inner workings of the 8086 (used by the AT&T 6300) and 8088 (IBM PC) chips...and describes specific techniques for using the full capability of these chip designs while programming in assembler. \$18.95

Now at your book or computer store.
Or order toll-free today:

800-624-0023

In New Jersey:
800-624-0024

Brady COMMUNICATIONS COMPANY, INC.
c/o Prentice Hall,
P.O. Box 512, W. Nyack, NY 10994

Circle the numbers of the titles you want below.
(Payment must be enclosed; or, use your charge
card.) Add \$1.50 for postage and handling.
Enclosed is check for \$_____ or charge to
☐ MasterCard ☐ VISA.

1 (0-89303-510-6)
5 (0-89303-575-0)

2 (0-89303-660-9)
6 (0-89303-484-3)

3 (0-13-917568-7)
7 (0-89303-584-X)

4 (0-89303-599-8)
8 (0-89303-424-X)

Acc't # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

(New Jersey residents, please add applicable sales tax.)
Dept. 3

GLDDJ-AO(9)

CACHE22 + CP/M 2.2 = CP/M Max!

CACHE22 is a front-end system program that buries all of CP/M 2.2 in banked memory. It helps 8080/Z80 computers to survive by providing up to 63.25K of TPA, plus the ability to speed disk operations, eliminate system tracks, and run Sidekick-style software without loss of transient program space. Complete source and installation manual, \$50.00.

CP/M is a trademark of Digital Research Inc.
Sidekick is a trademark of Borland International



MIKEN OPTICAL COMPANY

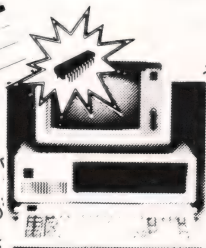
53 Abbett Avenue, Morristown, NJ 07960
(201) 267-1210



Circle no. 261 on reader service card.

PUT A CP/M COMPUTER IN YOUR PC!

RUN/CPM™



ARROW CROMENCO
QUME KAYPRO OS
RO OSBORNE TELEVI
MORROW CROMENCO
CO QUME KAYPRO

ONLY \$99.95
INCLUDING HARDWARE!
Plus over 20
CP/M80 programs!

And run 1,000's of CP/M programs
up to 30% faster, directly from your CP/M disks!

How does it work?

RUN/CPM virtually transforms your PC into any of the most popular CP/M systems. A simple replacement of your PC's 8088/86 microprocessor with our N.E.C. V-20/30 microprocessor gives your computer the ability to run both 8 bit CP/M and 16 bit MS-DOS programs. RUN/CPM will

transform your PC's floppy drives into CP/M drives able to directly read, write, and format over 100 CP/M disks! Terminal emulation supporting dozens of the most popular terminals completes the transformation of your PC into a CP/M system.

Performance?

Depending on the application, many of your CP/M programs will run up to 30% faster on your PC. Other features include: ability to run CP/M programs in color, logical and physical drive assignments, run CP/M or MS-DOS programs from the same prompt! RUN/CPM is the solution to running CP/M software on PC's.

To order
send check or money
order (U.S. funds)
add \$5.00 shipping/handling.

Micro Interfaces Corporation
6824 N.W. 169th Street, Miami, Florida 33015
(305) 823-8088
Telex: 5106004680 MICRO INTER CO
Ask About Our Intel Operating System Interfaces
OEM, VAR, Dealers, Inquiries Invited

ORDERS ONLY



1-800-637-7226



Trademarks: CP/M (Digital Research Inc.), IBM (IBM Corp.), RUN/CPM (Micro Interfaces Corp.), MS-DOS (Microsoft Inc.), INTEL (Intel Corp.)

Circle no. 110 on reader service card.

INFERENCE ENGINE

(Continued from page 38)

\$(SETQ L '(A B C D))

(A B C D)

\$L

(A B C D)

\$(CAR L)

A

\$(CDR L)

(B C D)

\$(SETQ Q '((A B C) D))

((A B C) D)

\$Q

((A B C) D)

\$(CAR Q)

(A B C)

\$(CDR Q)

(D)

Now that we can take a list apart, how can we put one together? The *CONS* function allows us to build a list out of its component parts. For example:

\$(CONS 'A 'B)

(A . B)

This example is noteworthy because of the dot between the A and the B. In this case, the list is terminated in a strange way: a normal list is terminated with *NIL* as the last element, but it doesn't print out that way. Here the last element is B. The dot is used to show this. To build a normal list, we must terminate with *NIL*. For instance:

\$(CONS 'A 'NIL)

(A)

\$(CONS 'A '(B))

(A B)

\$(CONS '(A B) '(C D))

((A B) C D)

\$(SETQ U '(X Y Z))

(X Y Z)

\$(SETQ V '(A B C))

(A B C)

\$(SETQ W (CONS U V))

((X Y Z) A B C)

\$(CAR W)

(X Y Z)

\$(CDR W)

(A B C)

Thus we can see that *(CONS (CAR L) (CDR L))* is just *L* itself.

Let's try writing a function of our own. We will take the factorial function. Factorial (*n*) is defined as the product of all the numbers from 1 to *n*, so factorial (1)=1, factorial (2)=2, factorial (3)=6, factorial (4)=24, and so on. *DEFUN* is a LISP-supplied function that defines functions. The following code implements a factorial function:


```
(DEFUN FACT (LAMBDA (N)
  ((ZEROP N) 1)
  (TIMES N (FACT (SUB1 N))) ))
```

The *LAMBDA* expression above is actually used to define the function, and *DEFUN* assigns this function the name *FACT*. *LAMBDA* can also be used to define a function that is used only once and not give that function a name. Refer to any good text on LISP for more information about this strange phenomenon.

The second line of *FACT* introduces us to a new construct. The double nesting of parentheses means that the *CAR* of that line is a predicate, or expression that evaluates to true or false. False is *NIL*, and true is anything that is not-*NIL*. There is a special atom for the name of true, just like *NIL* is the name for (). The name for true is the atom *T*. *ZEROP* is a predicate that tests for a value of zero. If in the second line $N=0$, then in the third line *FACT* will return a value of 1; otherwise, *FACT* will return a value of $N * FACT(N - 1)$. The *SUB1* returns the number that is the decrement of its argument.

The definition for *FACT* is recursive: it is defined in terms of itself. More conventional languages would probably encourage the programmer to use an iterative definition, such as the BASIC program:

```
10 INPUT N
20 FACT=1
30 FACT=FACT*N
40 N=N-1
50 IF N<>0 THEN GOTO 30
60 PRINT FACT
```

LISP allows for iterative constructs also. We could have written *FACT* as follows:

```
(DEFUN FACT (LAMBDA (N)
  (SETQ F 1)
  (LOOP
    (SETQ F (TIMES F N))
    (SETQ N (SUB1 N))
    ((ZEROP N) F) ))
```

The *LOOP* function repeats the evaluation of each of its elements in turn, until a conditional expression is satisfied, at which time the loop exits. The iterative form of a simple loop such as this will usually run faster than its recursive counterpart, but some problems, such as we will see in the unification algorithm later on, demand recursion. It is hard to imagine anyone who would find the iterative definition easier to understand than the recursive definition; it is very much like the verbal definition. Furthermore, it naturally handles the case for *FACT(0)*, which requires an extra *IF* in the iterative routine.

The muLISP programming environment, muSTAR, includes a built-in full-screen editor and pretty printer and a complete interactive debugger and program-monitoring package. I started using muLISP with the muLISP-80 version on 8-bit CP/M and have used muLISP-82 on the IBM PC, and I am now using muLISP-83 on an IBM PC.

DATALIGHT C

LARGE MODEL

The **Datalight C** compiler is a full-featured UNIX System 5 compiler for PC and compatible computers. Datalight C features fast compile time, DLC one-step compile command, a MAKE program, full 8087 and software floating point, and Lattice C compatibility. The library contains UNIX standard functions and PC specific functions with easy access to DOS/BIOS functions. The package includes source code for UNIX-like tools: cat, fgrep, diff wc, and pr.

\$60

The **Developer's kit** contains all features of the Datalight C compiler plus support for LARGE MEMORY MODEL, ROMable code, third-party debuggers and the complete source for all library functions and start-up code.

\$99

Datalight C provides tight, fast, production-quality code as shown in the following table (excerpt from "The State of C," PC-TECH Journal, January 1986, used with permission).

	Datalight	ECO	Lattice	Microsoft	Mark Wms.
Fibonacci	21.8	22.7	24.9	25.6	27.5
Sieve	21.0	24.8	21.0	26.3	23.3
Getc / Putc	24.0	73.9	62.2	51.3	62.2

Datalight

11557 8th Ave. N.E.
Seattle, Washington 98125
(206) 367-1803

VISA and MasterCard accepted. Add \$3 shipping (\$5 UPS BLUE). Outside USA add \$15. Washington State residents add 7.9% sales tax.

Requires MS-DOS 2.0 or later, 2 DSDD disks.

Lattice C, a trademark of Lattice Corp. MS-DOS, a trademark of Microsoft. UNIX is a trademark of Bell Labs.

Circle no. 203 on reader service card.

The LIBRARIAN Inc.

SOURCE INCLUDED

has taken the drudgery out of generating plots. These libraries contain the most complete collection of plotting subroutines you can buy. Each collection places over 40 routines (in FORTRAN or C language*) at your fingertips. The library makes it easy to:

- plot semi-log and log-log graphs
- draw arcs and circles
- scale charts to exact sizes
- label text in different sizes, directions, and orientations
- design bar charts and pie charts
- and much more

Each package includes source code and a comprehensive manual.

Operating Systems: MS-DOS and PC-DOS

Plotters Supported: Houston Instruments
Hewlett-Packard
HPGL compatibles

We are constantly testing our libraries on different computers and with different compilers including: Microsoft FORTRAN, DR FORTRAN-77, RM FORTRAN, Lahey F77L, CI-C86™, and MWC 86. Specify compiler when ordering.

Price: \$295.00

For More Information: The Librarian, Inc.
10435 Greenbough, #110
Stafford, TX 77477
(713) 499-7662 or 491-2088

*We're developing libraries in BASIC and Pascal, too!

Circle no. 117 on reader service card.

A Cellular Automaton Written in Expert-2

by Jack Park

*The ultimate idea is
that each cell could be
an individual
computer.*

It's been nearly two years since I first wrote the simple expert system for predicting local weather.¹ Since then, many individuals have taken to experimenting with Expert-2, and some interesting things have happened along the way. For openers, Expert-2 was originally published as a learning tool² that would be available to users for experimentation and perhaps for writing their own versions implementing inference engines considerably improved over the original Expert-2.

But that hasn't stopped some from doing important things with the tool. One group, for example, has written an expert system for diagnosing user problems with a satellite. Another individual has begun developing experimental medical diagnostic tools with Expert-2.

In the meantime, my closet affection for weather prediction persists, so I became interested in systolic arrays for taking grid data, performing blazing calculations, and pontificating on future events. It turns out that such a systolic array is easy to visualize as a cellular automaton, an array of cells that communicate with their nearest neighbors.

Inserting intelligence into each cell seems a logical point of departure, the idea being that "smart" cells could at first use knowledge provided by outside experts to begin issuing pronouncements. Later, such cells could implement learning strategies for improving their predictions.

Imagine the earth as a globe covered with hexagonal-shaped grid lines. Each cell enclosed by a set of grid lines could be 100 miles across.

Choose any size and shape cell. If you choose an octagon, the cell will communicate with eight neighbors. The octagon is the shape used in the program presented here.

The ultimate idea for the systolic array model is that each cell could, in theory, be an individual computer, complete with its own inference program, database, and knowledge base along with, say, nine communications ports. That's eight ports for nearest-neighbor linkage and one extra to send out the individualized results. In a more common systolic configuration, the final outputs travel across the array in so-called systolic waves until they reach an edge. In the array imagined for weather prediction, there would be no edge. That's just one candidate architecture for the weather prediction task.

That architecture, at least for weather prediction, seems reasonable, especially when you consider that weather in any given cell is affected by the weather of the nearest neighbor cells as well as the weather within the cell itself. Knowledge about that weather may be cell-specific, just as one considers the differences between polar and tropical environments.

In any event, the program listed with this article (Listing One, page 74) models the action of a cellular automaton as best a lone Von Neumann computer chip can. It is not modeled as a systolic array but as a *do* loop that treats each cell individually in a fashion similar to the way in which a multiprocessor systolic array would treat each cell. It's only a model. John Conway's game of Life was chosen for this test on the basis that it is well documented, never ceases to please, and is quite easy to understand.

The Game of Life

Because Life is so well documented, I won't go into how it works other than to point out that each cell represents an entity that is either alive or not alive. The state of each entity is determined by a group of rules that use the states of each individual's nearest neighbors as parameters. Thus, the game models an ecological system of sorts and as such is a reasonable trial model along the path of developing more complex modeling systems.

The prime issue developed by the program is the interface between the numeric aspects of the model—in this case, the simple counting of living nearest neighbors and the knowledge-based intelligence that guides the history of a cell. This program illustrates the repetitive calling of the inference system from inside a *do* loop that executes 256 times per display pass. With this technique it is possible to couple knowledge to procedural activities on a repetitive basis and effectively watch what a systolic array system would otherwise do quite quickly.

Life gives us a chance to study the performance of an inference engine. Expert-2 takes about 28 seconds to exercise a single pass. The cell-counting

Jack Park, P.O. Box 326, Brownsville, CA 95919

procedure takes about 2 seconds, leaving about 26 seconds for the inference mechanism to run 256 times. An experimental inference engine of a completely different design much closer to native Forth takes about 4 seconds per pass; with the same 2-second numeric part, that's about 2 seconds for the inference part. This statistic is interesting because it gives some evidence of Expert-2's absolute performance: the experimental inference engine benchmarks at about 2,700 logical inferences per second and runs Life about 13 times as fast as Expert-2 does. The implication is that Expert-2 runs about 200 LIPS, which isn't all that fast—so you tell your friends "it's pensive."

There's more to the timing than that; for example the newer inference engine has added features, such as disjunctive clauses (those coupled with *OR* or *ORNOT*). If users add such clause compilers to their Expert-2, fewer, more powerful rules could be written for the cell knowledge base, and the program would run faster. So it turns out that inference speed is partly a function of the inference engine algorithm and partly a function of the knowledge base itself. I leave it as an exercise for Expert-2 experimenters to see just how fast they can get the inference part of this program to run.

There's another thing I leave to experimenters who want to try running Life on their Expert-2: it turns out that the inference engine will happily write all over the video what it deduces on each of the 256 passes per epoch. That, of course, is unacceptable, especially because it wipes out the pretty grid display. So a feature must be added to Expert-2 that suppresses any deduction or conclusion printing. This feature is set to the suppress mode by the word *NOSHOW* and defaults to *SHOW*.

Perhaps one of the more striking features of the program is the separation of knowledge from all the procedural stuff. The rules listed between the Forth words *RULES* and *DONE* completely capture all the knowledge required for determination of the next state of any cell. This separation makes it a very simple matter to rewrite the rules governing the life history of individual cells and examine the impact of those revised rules

on the ecology of the small closed world.

One other point is worth considering: the world this exercise models is a 16-by-16 array, with the respective edges connected such that the world looks like a torus. This shape makes for some strange edge effects. What you see on the display is just the flat "apparent" world. You can easily increase the array size and leave the edges free or otherwise change the model. A 24-by-24 array would easily fit on the display of most computers.

The entire program compiles on top of Expert-2, which, of course, is compiled on top of Forth. Expert-2 will automatically separate the colon-defined procedural stuff from the knowledge base at the end of the listing.

A Single-Hypothesis Method

This implementation of a knowledge base illustrates a single-hypothesis method of forcing the inferencing process. In short, Expert-2 tries to prove—by the backward chaining method—some goal hypothesis. In this case there is only one hypothesis, *cell propagates*. In order to prove that hypothesis, the last rule shows that the inference engine must first prove the cell does not live, then that it does not die. If it neither lives nor dies, then it continues on (either living or not living). Suppose, however, that the counts are such that the cell lives (remember that this whole inference procedure runs once for each of 256 individual cells). If the cell lives—as, for example, if the first rule happens to fire—then the computer deduces *cell lives* but neglects to tell you it deduced that (remember *NOSHOW*), thus firing the *ANDTHEN-RUN LIVE* consequent clause and causing the hypothesis rule to fail. At that point, Expert-2 would be happy to tell you it cannot conclude anything, but amnesia again sets in because of *NOSHOW*. Then off to the next cell.

A key point to notice here is that Expert-2 wants a symbolic reference on which to base its inferences. The symbolic reference written into this program is the *THEN* clause that states the rule's consequence (e.g., *cell lives*, *cell dies*, etc.). Such string clauses make the rules quite readable. Furthermore, when combined

IBM COMPATIBILITY at a not so IBM price



TECH PC/AT \$1999

PRICE INCLUDES:

- 6MHz 80286 CPU
- 512K
- One, 1.2 MB Floppy Drive
- 8 Expansion Slots
- 195 Watt Power Supply
- Complete MS DOS, PC DOS, Xenix Compatibility
- Runs Lotus 123, dBase III Framework and all other popular AT Software.
- ONE YEAR WARRANTY!!

OPTIONS:

Tech PC/AT with 20MB Hard Disk **\$2399**

Tech PC/AT with 20MB Hard Disk, Monochrome Monitor, Hercules® Compatible Mono/Graphics Card **\$2599**

Also available with 6-8 MHz Switchable CPU, Tape Backups, Modems, Large Hard Disks, and Networking Systems.

TECH TURBO PC/AT ... \$2399

6-8 MHz Switchable 80286 CPU

TECH TURBO PC/XT ... \$1099

PRICE INCLUDES:

- 4 to 7 MHz Software Switchable CPU
- 640K
- Two, 360K DS/DD Floppy Disk Drives
- 8 Expansion Slots
- 135 Watt Power Supply
- ONE YEAR WARRANTY!!

OPTIONS:

Tech Turbo PC/XT with 20MB Hard Disk **\$1699**

Tech Turbo PC/XT with 20MB Hard Disk, Monochrome Monitor and Hercules® Compatible Mono/Graphics Card **\$1899**

TECH PC/XT \$799

PRICE INCLUDES:

- 4.77 MHz CPU
- 256K
- Two, 360K DS/DD Floppy Drives
- 8 Expansion Slots
- 135 Watt Power Supply
- ONE YEAR WARRANTY!!

OPTIONS:

Tech PC/XT with 20MB Hard Disk **\$1399**

Tech PC/XT with 20MB Hard Disk, Monochrome Monitor, Hercules® Compatible Mono/Graphics Card **\$1599**

TELEX: 272006

Answer Back-TECH

FAX: 714/556-8325

Visa, MasterCard, Check Accepted

TECH PC PERSONAL COMPUTERS

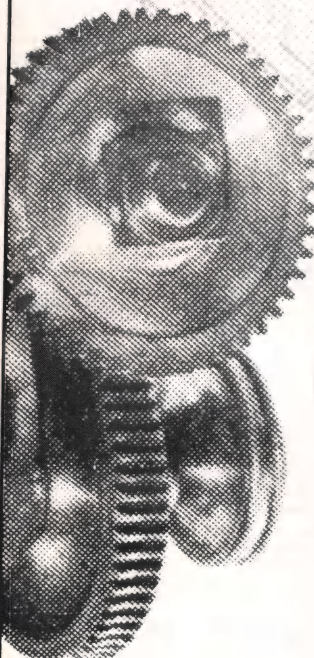
714/754-1170

2131 S. HATHAWAY, SANTA ANA, CA 92705

©1985 TECH PC
*Hercules is a registered trademark of Hercules Computer Technology.

*IBM, IBM PC, XT and AT are registered trademarks of International Business Machines Corp.

Dealers only circle no. 245 on reader service card.
Users only circle no. 279 on reader service card.



BD Software, Inc., maker of the original
CP/M-80 C Language Development
System, knows

Time is precious

So the compilation, linkage and execution speeds of BDS C are the fastest available, even (especially!) on floppy-based systems. Just ask any user! With 15,000+ packages sold since 1979, there are *lots* of users . . .

New! Ed Ream's RED text editor has been integrated into the package, making BDS C a truly complete, self-contained C development system.

Powerful original features: CDB symbolic source-level debugger, fully customizable library and run-time package (for convenient ROM-ing of code), XMODEM-compatible telecommunications package, and other sample applications.

National C User's Group provides direct access to the wealth of public-domain software written in BDS C, including text editors and formatters, BBS's, assemblers, C compilers, games and much more.

Complete package price: \$150.
All soft-sectored disk formats, plus Apple CP/M, available off-the-shelf. Shipping: free, by UPS, within USA for *prepaid* orders. Canada: \$5. Other: \$25. VISA, MC, COD, rush orders accepted.

BD Software, Inc.

BD Software, Inc.
P O Box 2368
Cambridge MA 02238
617 • 576 • 3828



EXPERT-2 (Continued from page 43)

with the numeric or procedural clauses (e.g., *ANDRUN COUNT=2*) such symbolic clauses create one way an inference engine can combine powerful symbolic inferencing with numeric processing. Of course, you could rewrite the rule compiler to allow in-line code—that is, to allow you to write into the rule the equation you want solved. I have chosen to separate numeric and symbolic representations, suspecting there might be some code economies caused by multiple calls to the same (or similar) procedures from different rules. Thus, the rules simply trade on either symbolic clauses or name fields of colon-defined procedures. This leads to some wild speculations on potential nonmonotonic reasoning strategies where one of the callable procedures is none other than *DIAGNOSE*, the main inference routine—which is something like your mutt chasing its tail.

Included is a pair of pattern initialization words: *EATER* and *PENTA*;³ they are supposed to act as oscillators. You can also add other initializing patterns. You must initialize an array before running it by typing one of the patterns and then typing the word *RUN* to start the whole works off. Tap the space bar to kill a run—it will stop after the next display or after 32 cycles. Enjoy.

Notes

1. J. Park. "Expert Systems and the Weather," *DDJ* 90 (April 1984).
2. J. Park. *Expert Toolkit* (Mountain View, Calif.: Mountain View Press, 1984). Also available from Parsec Research and Miller Microcomputer Service.
3. David Buckingham. "Some Facts of Life," *Byte* (December 1978).

DDJ

(Listing begins on page 74.)

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service No. 3.

NEW RELEASE

Ecosoft's Eco-C88 Rel. 3.0 C Compiler



\$59⁹⁵

Release 3.0 has new features at an unbelievably low price. ECO-C88 now has:

- Prototyping (the new type-checking enhancement)
- enum and void data types
- structure passing and assignment
- All operators and data types (except bit fields)
- A standard library with more than 200 functions (many of which are System V compatible for greater code portability)
- cc and mini-make that all but automates the compile process
- 8087 support (we sense the 8087 at runtime – no dual libraries)
- ASM or OBJ output for use with MSDOS linker
- Tiered error messages – enable-disable lint-like error checking
- Fast compiles and executing code
- Expanded user's manual
- Enhanced CED program editor (limited time offer)

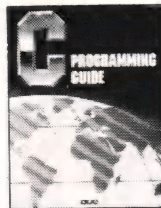
We also offer the following support products for Eco-C88.

CED Program Editor

\$29⁹⁵

CED now supports on-line function help.

If you've forgotten how to use a standard library function, just type in the name of the function and CED gives you a brief summary, including function arguments. CED is a full screen editor with auto-flagging of source code errors, multiple windows, macros, and is fully configurable to suit your needs. You can edit, compile, link, and execute DOS commands from within the editor. Perfect for use with Eco-C88. For IBM PC, AT and look alikes.



C Programming Guide \$20

After reading the 1st edition, Jerry Pournelle (BYTE Magazine) said: "I recommend this book ... Read it *before* trying to tackle Kernighan and Ritchie." The second edition expands this best seller and walks you through the C language in an easy-to-understand manner. Many of the error messages include references to this book making it a perfect companion to Eco-C88 for those just starting out with C.

C Source for Standard Library

\$10

Contains all of the source code for the library functions that are distributed with Eco-C88, excluding the transcendental functions written in assembler.

(\$20 if not with order)

Developer's Library

\$25

Contains the source code for all library functions, including the transcendental functions and those written in assembler. Perfect for the developer that wish to write their own custom functions or learn how we implemented the Eco-C88 library.

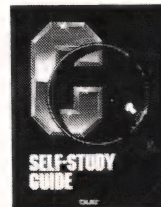
(\$50 if not with order)

ISAM Library

\$15

Contains the code from the C Programmer's Library in relocatable format (i.e., .OBJ) including the delete code for the ISAM file handler.

(\$30 if not with order)



C Self-Study Guide \$17

(Purdum, Que Corp.). Designed for those learning C on their own. The book is filled with questions-answers designed to illustrate many of the tips, traps, and techniques of the C language. Although written to complement the Guide, it may be used with any introductory text on C.



C Programmer's Library \$20

(Purdum, Leslie, Stegemoller, Que Corp.). This best seller is an intermediate text designed to teach you how to write library functions in a generalized fashion. The book covers many advanced C topics and contains many useful additions to your library including a complete ISAM file handler.

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later. Call today:

1-800-952-0472 (for orders)

or

1-317-255-6476 (tech. info.)



Ecosoft, Inc.
6413 N. College Ave. • Indianapolis, IN 46220



TRADEMARKS: ECO-C88, ECOSOFT

Modeling a System in PROLOG

by Sheldon D. Softky

This article discusses the difficulty of specifying large software systems and how you can model them on a personal computer if you use the right language. Software now costs much more than its hardware if the system is really large. The Department of Defense is so concerned about this problem that it has instituted a program to tackle it.¹

The biggest difficulty seems to be getting the specifications right for a system so that what is built is really what is wanted. Experience shows that you must first find answers to the following questions:

- What do the users want to input?
- How do they want to exercise control?
- What do they want the output to look like?
- What kind of logical relations are needed to support the above needs?
- Do the obvious processing goals really satisfy the stated requirements?

What is needed to answer the above questions properly is an executable model of the entire system. This would be, in the strictest sense of the word, a functional specification. It would allow the users to get a hands-on feel of how the system serviced them; even more important, it would be complete enough to verify the adequacy of the requirements and logical structure needed. Current thought about software development seems to be focusing on exe-

How do you produce a program that runs like the real system without being the real system?

cutable specifications as necessary tools for effective development.²

How do you produce a computer program that executes like the real system does without its being the real system? The idea is to save mistakes (and money) by getting a trial run of the whole system's important features, but it must be done cheaply. Using rapid prototyping has become popular because some features needed in the final system are not needed in a prototype. For example,

- The prototype doesn't have to be as fast as the intended product.
- It doesn't need to handle the intended full-size database.
- It doesn't need to be in the target language or on the target machine.
- It doesn't have to be free from errors or contain provision for error management.

Because these features are unnecessary, it is possible to model a large system cheaply.

Modeling an entire system calls for something extra by way of representing the specification—namely, a specification language created expressly for this purpose. Several languages are available, but only very re-

cent ones produce specifications that can be executed. I will show here how the well-known logic programming language PROLOG has many features that make it ideal for expressing such executable specifications.

Advantages of PROLOG

It's desirable that you be able to read a specification without a week's training course to learn a new language. I shall show, in an example, how simple the format is for micro-PROLOG from Logic Programming Associates. Any engineer or programmer should be able to read a modest PROLOG program after studying the language for an hour or two. PROLOG stands for "programming in logic," and it's not surprising that it shows clearly the logical structure required for a system.

PROLOG also acts as an aid to design. This follows immediately from the form of its sentences (rules or facts). The structure of the language actually encourages the usual hierarchical decomposition of functions into subfunctions. When a specification becomes unwieldy to express in PROLOG, maybe the wrong thing is being attempted.

A third advantage of the language is that it helps to demonstrate I/O. It has features that provide both output, as part of language sentences, and the ability to request input as a consequence of execution. The most commonly used form of output is answers to queries about the contents of a database.

In terms of cost-effectiveness, PROLOG couldn't be better. The edition I am using costs less than \$300 (on a disk) and runs on a computer that costs less than \$700 (Osborne 1).

It is important that an executable

Sheldon D. Softky, ABC Press of Silicon Valley, 320 Encinal Ave., Menlo Park, CA 94025

specification does not tempt customers to use it as a cheap substitute for the real software product. The consequences of this happening are disastrous; they frequently discourage any use of prototyping that might be mistaken for part of the final product. The advantage of a cheap PROLOG that runs on a cheap computer is that it is unlikely to support either the speed or the size of a real-world information system and thus the resulting prototype could not be mistaken for the final product.

PROLOG's weakest point is the way in which it models the intended environment. Any simulation of the target implementation environment would have to be provided in the logical structure of the program (which is quite possible but must be done by the designer).

A System Described in PROLOG

I have chosen to model a hospital administrative information system. Greenspan, Mylopoulos, and Borgida³ used such a system to demonstrate their custom-built specification language, RMF. I am indebted to them for suggesting this type of system and for the stimulation that led to PROLOG usage. The suggestion that PROLOG would be good for stating specifications was actually made by Shapiro,⁴ but I haven't seen any implementation of the suggestion.

Most information systems must store a common core of reference data about their subjects. The primary subject of the information system in this example is the hospital patient; the basic core of reference data for a patient certainly includes

- Last name, first name, initial
- Social security number
- Address: street and number, city, state, ZIP
- Phone

Some basic ID number must be chosen for a subject; I have chosen the social security number as the primary identifier for a hospital patient. All other data items are keyed to this one for an individual patient. I make each of the simplest items above into a single fact about the patient and give it a PROLOG name that shows the function of the data item. This functional

name is called a predicate. The above facts are represented in the form

(predicate data-value ID-value)

which for a patient named Humpty Dumpty could be:

```
(social-sec-no S4064)
(firstname-ini Humpty S4064)
(lastname Dumpty S4064)
(housenumber H4701 S4064)
(street Broadway S4064)
(city Provo-Utah S4064)
(zip P89201 S4064)
(phone 405 "—" 1191 S4064)
```

PROLOG actually allows any number of data values or ID values to be contained in a fact, following the predicate. You can pull together several facts about the same patient by defining a rule that enumerates those facts, with data items mentioned as variables instead of as constants in the referenced facts. A PROLOG rule defines the relation *person-id* as the simultaneous matching of facts about a patient: *firstname-ini*, *lastname*, and *social-sec-no*, as shown in Table 1, page 48. Note that the common variable name in all the facts is *social-sec-no*. Note also that I

It's 3 AM !



Do you know where your bugs are ?

This C programmer is finding his bugs the hard way ... one at a time. That's why it's taking so long. But there's an easier way. Use

PC-Lint

PC-Lint* analyzes your C programs (one or many modules) and uncovers glitches, bugs, quirks, and inconsistencies. It will catch subtle errors before they catch you. PC-Lint resembles the Lint that runs on the UNIX* O.S., but with more features and some awareness of the 8086 environment.

- Full K&R C
- Supports Multiple Modules—finds inconsistencies between declarations and use of functions and data across a set of modules comprising a program.
- Compares function arguments with the associated parameters and complains if there is a mismatch or too many or too few arguments.
- User-modifiable library description files for most major compilers.
- All warning and information messages may be turned on and off globally or locally (via command line and comments) so that messages can be tailored to your programming style.
- All command line information can be furnished indirectly via file(s) to automate testing.
- Use it to check existing programs, programs about to be exported or imported, as a preliminary to compilation, or prior to scaling up to a larger memory model.
- All one pass with an integrated pre-processor so it's very fast.
- Has numerous flags to support a wide variety of C's, memory models, and programming styles.
- Price: \$139.00 MC, VISA (Includes shipping and handling) PA residents add 6% sales tax. Outside USA add \$10.00
- Runs under MS-DOS* 2.0 and up, with a minimum of 128Kb of memory. It will use all the memory available.

Trademarks: PC-Lint (Gimpel Software), UNIX AT&T, MS-DOS (Microsoft).

NEW !!

Amiga - Lint
Special Introductory Price
\$98.00

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

MODELING IN PROLOG (Continued from page 47)

have used imitation social security numbers of only four digits as a convenience.

The only thing you have to learn to write such PROLOG sentences is the meaning of the punctuation. The whole sentence must be enclosed in outer parentheses, the relation clause being defined by the rule must be the first clause and be enclosed in parentheses, and each other clause that must be matched

for the rule to be satisfied must also be enclosed in parentheses. The first item appearing in each clause is the name of that fact or rule, and is called a relation (or predicate). There is an implied IF after the relation clause being defined and implied logical ANDs between all the clauses that support the relation. Thus you would read the rule in Table 1 as: *(person-id X Y Z) IF (lastname Y Z) AND (firstname-ini X Z) AND (social-sec-no Z).*

The power of PROLOG rules is that a rule can reference other rules as well

```
((person-id X Y Z)
  (lastname Y Z)
  (firstname-ini X Z)
  (social-sec-no Z))
((firstname-ini B S123))
((firstname-ini Tom-T S3475))
((firstname-ini Mrs-Jack-S S4831))
((firstname-ini Humpty S4064))
((firstname-ini Don S4776))
((firstname-ini Don S4895))
((lastname A S123))
((lastname Thumb S3475))
((lastname Spratt S4831))
((lastname Dumpty S4064))
((lastname Giovanni S4776))
((lastname Quixote S4895))
((social-sec-no S123))
((social-sec-no S3475))
((social-sec-no S4831))
((social-sec-no S4064))
((social-sec-no S4776))
((social-sec-no S4895))
```

Table 1

```
((address X Y Z x y z X1 Y1 Z1)
  (person-id X Y Z)
  (hounumber x Z)
  (street y Z)
  (city z Z)
  (zip X1 Z)
  (phone Y1 "—" Z1 Z))
((hounumber C S123))
((hounumber H14085 S3475))
((hounumber H1808 S4831))
((hounumber H4701 S4064))
((hounumber H3851 S4776))
((hounumber H21105 S4895))
((street D S123))
((street Abelson-St S3475))
((street Rose-Way S4831))
((street Broadway S4064))
((street Alameda S4776))
((street El-Camino S4895))
((city E S123))
((city San-Carlos S3475))
((city Burlingame S4831))
((city Provo-Utah S4064))
((city Menlo-Park S4776))
((city Palo-Alto S4895))
((zip G S123))
((zip P94065 S3475))
((zip P94072 S4831))
((zip P89201 S4064))
((zip P94025 S4776))
((zip P94043 S4895))
((phone 923 "—" 1436 S123))
((phone 364 "—" 2915 S3475))
((phone 291 "—" 1507 S4831))
((phone 405 "—" 1191 S4064))
((phone 329 "—" 51 S4776))
((phone 326 "—" 8195 S4895))
```

Table 2

Programmer Essentials

"Offers many capabilities for a reasonable price"

W. Hunt, PC Tech Journal

"I highly recommend the C UTILITY LIBRARY"

D. Deloria, The C Journal

ESSENTIALS

200 functions: video, strings, keyboard, directories, files, time/date and more. Source code is 95% C. Comprehensive manual with plenty of examples. Demo programs on diskette. Upgrade to THE C UTILITY LIBRARY for \$95.

\$100

THE UTILITY LIBRARY

Thousands in use world wide. 300 functions for serious software developers. The C ESSENTIALS plus "pop-up" windows, business graphics, data entry, DOS command and program execution, polled async communications, sound and more.

\$185

ESSENTIAL GRAPHICS

Fast, powerful, and easy to use. Draw a pie or bar chart with one function. Animation (GET and PUT), filling (PAINT) and user definable patterns. IBM color, IBM EGA and Hercules supported (more soon). NO ROYALTIES. Save \$50 when purchased with above libraries. Available February, 1986.

\$250

Compatible with Microsoft Ver. 3, Lattice, Aztec, Mark Williams, CI-C86, DeSmet, and Wizard C Compilers. IBM PC/XT/AT and true compatibles.

C Compiler Packages: Microsoft C - 319, Lattice or CI-C86 compilers - \$329. Save \$40 - \$50 when purchasing compiler and library combinations. Specify C compiler and version number when ordering. Add \$4 for UPS or \$7 for UPS 2-day. NJ residents add 6% sales tax. Visa, MC, Checks, PO's.



ESSENTIAL SOFTWARE, INC

P.O. Box 1003 Maplewood, NJ 07040 914/762-6605

as facts. Thus complex relationships can be functionally decomposed by nesting rules within rules within rules and so on until, at the bottom level, only facts are referenced. A simple example of this is the definition of the relation *address* shown in Table 2, page 48, in which the relation *person-id* is used and all the rest of the clauses are simply facts.

There are some more complex conditions for entering a patient into a hospital database than those I have discussed so far. For example, several checks must be made before admission is complete:

- Is there room left in the hospital?
- Can the patient pay?
- Has the patient's personal data been entered?
- Has the patient been assigned to a ward?
- Has the patient been assigned both an attending physician and consulting physician?
- Does the specialty of one of these physicians match the type of ward that was assigned?

Although many other conditions are associated with entering a hospital, these suffice to demonstrate how PROLOG encourages hierarchical decomposition of the functions of a system. A clear expression of the above conditions requires some deeply nested rules, which can be arrived at bottom-up or top-down, as you wish.

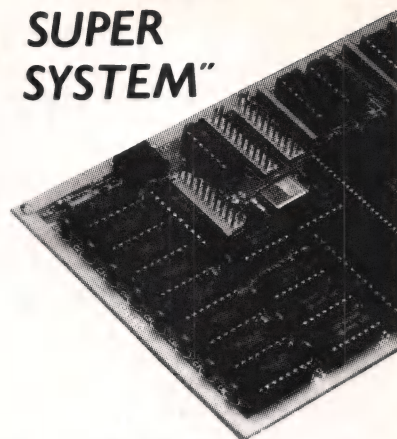
Table 3, below, shows these rules top-down. The relation *in-hospital* is true if there is a *person-id* and an *admit-date* but not a *release-date* or a *died-on* date. There is an *admit-date* for the patient if the rule for *admit-patient* has been satisfied. (ADDCL puts that clause in the database; DELCL removes the clause.) This is a deeply nested rule that uses a check for *room-is-left-for* and the data provided for *submit-patient*. *Room-is-left-for* counts the total patients if the admission takes place and verifies that the *person-id* isn't already admitted. *Submit-patient* checks the physician and ward assignments and whether the patient is financially responsible and provides reference to *current-date*. It refers to the *specialty-match* relation,

```
((in-hospital X Y Z x)
  (person-id X Y Z)
  (admit-date x Z)
  (NOT release-date y Z)
  (NOT died-on z Z))
(admit-patient X Y)
  (* admits patient after checking for space and assignments)
  (room-is-left-for X Y Z)
  (submit-patient Y x)
  (DELCL ((patient-count Z)))
  (ADDCL ((patient-count X)))
  (ADDCL ((admit-date x Y)))
(room-is-left-for X Y Z)
  (* checks new patient-count against available space, and whether)
  (* person not yet admitted, but personal data entered)
  (patient-count Z)
  (SUM Z 1 X)
  (room x)
  (OR ((LESS X x)) ((EQ X x)))
  (person-id y z Y)
  (NOT admit-date X1 Y))
(submit-patient X Y)
  (* checks ability to pay and MD, ward assignments)
  (can-pay X)
  (ward-name Z X)
  (atten-phys x X)
  (cons-phys y X)
  (specialty-match x y Z)
  (current-date Y))
```

Table 3

Byte Magazine called it.

"CIARCIA'S SUPER SYSTEM"



The SB180 Computer/Controller

Featured on the cover of Byte, Sept. 1985, the SB180 lets CP/M users upgrade to a fast, 4" x 7 1/2" single board system.

- **6MHz 64180 CPU**
(Z80 instruction superset), 256K RAM, 8K Monitor ROM with device test, disk format, read/write.
- **Mini/Micro Floppy Controller**
(1-4 drives, Single/Double Density, 1-2 sided, 40/77/80 track 3 1/4", 5 1/4" and 8" drives).
- **Measures 4" x 7 1/2"**, with mounting holes
- **One Centronics Printer Port**
- **Two RS232C Serial Ports**
(75-19,200 baud with console port auto-baud rate select).
- **Power Supply Requirements**
+5V +/-5% @500 mA
+12V +/- 20% @40mA
- **ZCPR3 (CP/M 2.2/3 compatible)**
- **Multiple disk formats supported**
- **Menu-based system customization**

SB180-1

SB180 computer board w/256K bytes RAM and ROM monitor
..... **\$369.00**

SB180-1-20

same as above w/ZCPR3, ZRDOS and BIOS source..... **\$499.00**

-Quantity discounts available-

NEW

COMM180-M-S

optional peripheral board adds
1200 bps modem and SCSI
hard disk interface.

**TO ORDER
CALL TOLL FREE
1-800-635-3355**

**TELEX
643331**

For technical assistance or
to request a data sheet, call:

1-203-871-6170



**Micromint, Inc.
25 Terrace Drive
Vernon, CT 06066**

which checks whether either of the physicians assigned have a specialty matching the *ward-name*. It is desirable to express *specialty-match* in terms of *phys-spec-ward* and *phys-spec-cons*, which reference the physicians' specialties listed under *phys-spec*. (See Table 4, right.)

All these rules, no matter what their level in the decomposition, can be individually queried whenever desired. This is a big difference from a "standard" language, where a whole program must be executed to test anything. The ability to "peek into" a PROLOG specification makes it more manageable than is a conventional prototype, and it is equally easy to change something in a visible way and check that it worked properly there.

Execution of a PROLOG Specification

PROLOG executes as an interpreter—that is, it compiles and executes line by line (as does BASIC). Sentences typed in from the keyboard are added to the contents of the program workspace and contribute to the execution if they are referenced. The usual way in which you execute PROLOG is by typing in a query from the keyboard that references one or more rules or facts. The contents of a query can be a set of clauses with a mixture of variables and constants.

There are two forms of query: a yes/no answerable question and a question whose answer contains values for all the variables mentioned. If a single rule is referenced with none of its variable values specified, PROLOG replies with the values for all the variables that are requested. The PROLOG data used to answer queries consists of facts that are tried in rules that are referenced, plus values provided in the query itself. The data is substituted in all the supporting clauses for a rule, using the matched variable names for identical items. If a complete match is possible using only facts that are in the database, those values requested in the query are printed as an answer or yes/no for that type of query. The entire database is examined in a backward scan that tries all possible combina-

```
((can-pay X)
  (OR ((bluecross X REF-NO Y)) ((cred-ref X BANK-NAME Z))))
((specialty-match MD1 MD2 ward-spec))
((specialty-match X Y Z)
  (/ * checks that either ward MD or consulting MD has)
  (/ * specialty that matches the ward)
  (OR ((phys-spec-ward Z X)) ((phys-spec-cons Z Y))))
((phys-spec-ward ward-spec MD))
((phys-spec-ward X Y)
  (/ * specialty of the assigned ward MD)
  (phys-spec Z Y)
  (EQ Z X))
((phys-spec-cons cons-spec MD))
((phys-spec-cons X Y)
  (/ * specialty of the assigned consulting MD)
  (phys-spec Z Y)
  (EQ Z X))
((phys-spec specialty MD))
((phys-spec surgery John-Jones))
((phys-spec ear-nose-throat Bill-Smith))
((phys-spec dermatology Jim-White))
((phys-spec obstetrics Jim-Jackson))
((phys-spec pediatrics Bob-Black))
```

Table 4

```
all ((X Y Z) (person-id X Y Z))
(B A S123)
(Tom-T Thumb S3475)
(Mrs-Jack-S Spratt S4831)
(Humpty Dumpty S4064)
(Don Giovanni S4776)
(Don Quixote S4895)
No (more) answers
&.

all ((X Y Z x y z X1 Y1 Z1) (address X Y Z x y z X1 Y1 Z1))
(B A S123 C C E G 923 1436)
(Tom-T Thumb S3475 H14085 Abelson-St San-Carlos P94065 364 2915)
(Mrs-Jack-S Spratt S4831 H1808 Rose-Way Burlingame P94072 291 1507)
(Humpty Dumpty S4064 H4701 Broadway Provo-Utah P89201 405 1191)
(Don Giovanni S4776 H3851 Alameda Menlo-Park P94025 329 51)
(Don Quixote S4895 H21105 El-Camino Palo-Alto P94043 326 8195)
&.

all ((X Y Z x) (in-hospital X Y Z x))
(Tom-T Thumb S3475 D840815)
(Mrs-Jack-S Spratt S4831 D840815)
No (more) answers
&.

all ((X Y Z) (can-pay Z) (person-id X Y Z))
(B A S123)
(Tom-T Thumb S3475)
(Humpty Dumpty S4064)
(Don Giovanni S4776)
(B A S123)
(Mrs-Jack-S Spratt S4831)
(Don Quixote S4895)
No (more) answers
&.

ask ((specialty-match John-Jones Bill-Smith surgery))
yes
&.
```

Table 5

tions. When answers are exhausted, PROLOG prints "No more answers."

The type of query that prints values consists of *all*((list of some variables)/(some fact or rule)/(some other fact or rule)/(etc . . .)) with each variable in the list mentioned in at least one of the rules that are referenced. The yes/no query is "ask (a rule with values already substituted)/(another such rule)/(etc . . .)." Table 5, page 50, shows some queries of rules I have described in the hospital database.

Using PROLOG for a Specification Tool

Obviously the expression of a specification, such as the one in this example, doesn't replace the document you are already familiar with—it complements it. Most customers for large software systems have their established requirements for an initial specification that is written in text. The executable specification should be looked upon as a useful extension to prototyping and a help in experimenting with designs. This technique is so inexpensive that it should be accessible to everyone who develops software or studies systems.

Notes

1. Called the STARS Program (Software Technology for Adaptable, Reliable Systems). See *IEEE Computer*, Nov. 1983, for a review.
2. L. Brice, J. Connel, and D. Shafer, "Using INGRES as a Rapid Prototyping Development Tool," *Proc. IEEE Symposium on the Applications and Assessment of Automated Tools for Software Development* (Nov. 1983): 34.
3. S. Kamin, S. Jefferson, and M. Archer, "The Role of Executable Specifications," *ibid.*: 105.
4. B. Meyer, "On Formalism in Specifications," *IEEE Software* (Jan. 1985): 6.
5. S. Greenspan, J. Mylopoulos, and A. Borgida, "Capturing More Real World Knowledge in the Requirements' Specification," *Proc. IEEE Sixth International Conference on Software Engineering* (1982): 225.
6. E. Shapiro, "The Fifth Generation Project, a Trip Report," *Comm. of the ACM* (Sept. 1983): 637. **DDJ**

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 4.

LEARN LISP

Interactively and Write "Realistic" Programs with TransLISP for Only \$75

A "COMMON LISP" compatible Tutorial, Interpreter, Debugging, and Pretty Printer plus a Fast, Full Screen Editor, Samples and Help

☐ Start Easily and Quickly:

A complete, modular tutorial helps you learn LISP at your own pace. An integrated, interactive environment provides all of the elements needed to enter, modify, analyze and debug programs.

☐ Natural Language, Expert Systems and Mailing List:

Natural Language concepts are illustrated by a phone number retrieval program. Choose the best word processing program for you with the Expert System. File handling and typical data processing work are demonstrated by a Mailing List program.

☐ Write Realistic Programs:

Short examples and substantial programs of about 10 pages in length help you learn by modifying, studying and using the key concepts needed to write programs of 1000 lines or more.

☐ The "COMMON LISP" Standard: TransLISP includes a 230+ function subset of the "COMMON LISP" Standard. Use extras like the MSDOS interface and graphics. Or use "strict compatibility" to make programs written in TransLISP, with no changes, work with other COMMON LISP systems like VAX LISP, GC/LISP or LISP Machine LISP.

Use and Modify the Mailing List program to learn how to handle "normal" programming in LISP.

Runs on any MSDOS or PC DOS Systems: Not copy-protected, TransLISP is available in just about any 3", 5" or 8" format. PC compatibles can run TransLISP with no installation procedure. 192K memory and 1 floppy drive are the minimums required.

ONLY

\$75

Full refund if not
satisfied during
first 30 days.

For Beginners and Experienced Programmers

**Solution
Systems™**

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle no. 148 on reader service card.

Learn and Use AI Technology In Your First Evening With PROLOG-86

**NOW ONLY
\$95**

A complete *Prolog Interpreter, Tutorial, and set of Sample Programs:*

☐ Modify and write Expert Systems.

Use the simple "Guess the animal" example on the Tutorial or use the sophisticated system for Section 318 of the US Tax Code written by one of the PROLOG-86 authors and published in the March, 1985 issue of Dr. Dobb's Journal.

☐ Understand Natural Language

Use the sample program that produces a dBase DISPLAY command as output.

Programming experience is not required, but a logical mind is.

Prolog-86 Plus for \$250 adds: Windowing, 8087, 640K memory access, random access files, strings support and definite clause grammar.

RECENT IMPROVEMENTS: Floating point support, MSDOS commands, on-line help, load Editor.

AVAILABILITY: All MSDOS, PC DOS systems.

ONLY

\$95

Full refund if not
satisfied during
first 30 days.

**Solution
Systems™**

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle no. 152 on reader service card.

A 68000 Cross Assembler—Part 1

by Brian R. Anderson

The first thing I realized is that I would need some way to handle 32-bit integers.

I tend to be obsessive. For more than two years I have been admiring the Motorola 68000 from afar, wanting to do something with it but not knowing quite what. Although I've worked a bit with assembly language and with microprocessor hardware (Z80, Z8000, and M6800), I wasn't really what you'd call a "let's get close to the metal" type of hacker. What to do?

My love affair with Modula-2 started about a year ago. After coming up through FORTRAN, Pascal, and C (with a smattering of COBOL, BASIC, and even LISP thrown in), Modula-2 seemed an ideal language. Here was a high-level structured language that you could actually use to write operating systems, and unlike C, even make sense of the code afterward. I had it! I'd write a 68000 assembler in Modula-2. I was off to the races.

Bottoms Up?

Modula-2 is a great language for both top-down and bottom-up design. The top-down aspect is usually more highly touted, but often the immediate need is for some tool so that you can proceed with a design. The first thing I realized as I started to plan this project was that I would need some way to handle 32-bit integers. The 68000 uses operands that are up to 32 bits wide, whereas the implementation of Modula-2 that I was using (Hochstrasser's Modula-2 System for Z80 CP/M) provided only 16-bit integers, as the compiler was written before Wirth amended the language to include *LONGINT*, *LONGCARD*, and *LONGREAL*.

My first task, then, was to create a bottom-up module to handle the 32-

bit numbers I would need throughout the project. The LongNumbers module provides all the facilities to input, manipulate, and output a new data type that I called *LONG*, which acts essentially as an 8-digit hexadecimal number. Although I could have used assembly language or tricky machine-dependent low-level Modula-2 code to create a more efficient implementation, I decided to forego efficiency for portability because I plan to transport the assembler to other environments. (I have ported the assembler to the IBM PC using the Logitech compiler.)

Listing One, page 76, shows the definition module for LongNumbers. The type *LONG* is simply an array of *INTEGER*. I chose *INTEGER* instead of *CARDINAL* or subrange *[0..15]* to ease handling of carry/borrow in the arithmetic procedures. Most of the procedures are pretty straightforward, but some may need clarification. *CardToLong* and *LongToCard* provide conversions. This allows some flexibility so the assembler can accept 68000 addressing offsets (and even constants) in either hexadecimal or decimal. Because *CARDINAL* has a much smaller range than *LONG* has, not all conversions are possible—*LongToCard* returns *FALSE* in such cases. *StringToLong* converts a sequence of ASCII characters into a

LONG, returning *FALSE* if any illegal character is encountered. Like much of the code that I write now, *LongCompare* is patterned after a similar C routine for comparing strings. The two output routines, *LongPut* and *LongWrite*, are different from the rest of the routines in that they don't actually use type *LONG*; instead, they use open array parameters, which allows them to output an arbitrary sequence of hex digits. The last two routines, *AddrBoundW* and *AddrBoundL*, are needed because the 68000 insists that certain types of instructions and data begin at "even" addresses.

One other bit of bottom-up design occurred at the beginning of this project and resulted in a general-purpose library routine. I've always liked the way C handles command line arguments (with the standard parameters *ArgC* and *ArgV*). For those readers unfamiliar with C, *ArgC* is a count of the number of command line arguments encountered by the operating system, and *ArgV* is a pointer to those arguments. My module *CmdLin2* mimics this behavior for the Modula-2 environment. The definition module (Listing Two, page 76) shows *ArgV* as an *ADDRESS*; it is used in the main program as a *POINTER TO ARRAY OF POINTER TO STRING* in much the same way as C would use it. (Note: This is a machine-dependent module—CP/M-80 only! It assumes that the command line will be located at memory location 80H, with a count in the first byte. Programmers working in other environments will have to adapt at least the absolute addressing used in the implementation.)

Design Phase

With a few tools in hand and more

Brian Anderson, 2977 East 56th Ave.,
Vancouver, B.C. V5S 2a2 Canada

confidence than any believer in Murphy's Law has a right to have, I sat down to do a requirement analysis. I came up with the specifications shown in Table 1, page 54.

Jumping ahead just a bit: You might want to ask how well the final program adhered to these specifications. I fell short in a couple of areas and went beyond the original specifications in others. I never did implement the *RORG* assembler directive as a linker is required for it to be of any use and I haven't written a linker (yet!). Also, the assembler does not support binary constants. One additional pseudo-op (*EVEN*) is supported, however, and limited ASCII string evaluation was added. The error messages finally implemented are somewhat more extensive.

Implementation of X68000

The X68000 Cross Assembler is written in standard Modula-2, as defined by Niklaus Wirth in the second edition of *Programming in Modula-2*, (Springer-Verlag, 1983). The only possible machine dependency (aside from the *CmdLin2* module already mentioned) is because of the assumption that *INTEGERS*, *CARDINALS*, and *BITSETS* all occupy 16 bits of memory. Most microcomputer implementations and even several minicomputer implementations conform to this standard. Porting considerations will be mainly in the area of I/O library routines. The Hochstrasser library is virtually identical to the Volition Systems library, so little more than recompiling should be necessary for this popular compiler.

In the August 1984 issue of *Byte*, Wirth made a few comments about modules that are appropriate to any discussion of a major Modula-2 project:

"With the module we have added another level of granularity in program structuring. The difficulties of finding a good partitioning—I carefully avoid the word 'optimal'—are culminated at this level. . . . Lucky are those who hit a good solution at the outset, for any change affects all participant" modules.

Amen to that! My initial partitioning had a module that I called Parser doing the decomposition of source

lines into parts of speech, as well as syntax analysis and code generation. After the module had grown to more than a dozen large procedures and more than a thousand lines of code, I conceded that this was not the "optimal" partitioning. In the end, I split the original module into three smaller modules: Parser, SyntaxAnalyzer, and CodeGenerator. This splitting made for some rather awkward variable and type importations. With that disclaimer, we can go on to look at the data flow diagrams for the fin-

ished program.

Assembly—Pass 1

The purpose of the first pass through the 68000 source code is mainly to build a symbol table. As each instruction is scanned, an address counter is advanced based on the length of the instruction (68000 instructions vary in length from 2 to 10 bytes). When an *EQU* pseudo-op is encountered, its value must be entered in the symbol table, and when any other label is encountered, the value of the current



Finally! A library of high level C functions (not just a bunch of building blocks) designed to increase your productivity and help develop superior applications in dramatically less time! How? Well, Vitamin C automatically coordinates the complex tasks and leaves the programmer free to be creative! With Vitamin C, for example, you'll never even have to think about saving or restoring portions of the screen when a window is opened, closed or moved. Simply call *wopen()*, *wclose()* or *wmove()* and Vitamin C takes care of the complexities. It's just that easy! This philosophy of relieving the programmer from as many details as possible runs throughout Vitamin C. As a result, jobs that used to take days are finished in a matter of hours!

Includes 100% source, reference manual, step by step tutorial, examples, sample programs. Specify Microsoft v3, Lattice, Aztec, Computer Innovations, DeSmet or Mark Williams. Ask about UNIX, TI-Pro and other compatibility!

Vitamin C \$149.95

Perfect Windows + Powerful Data Entry already integrated !

Vitamin C's versatile features include...

- Complete input formatting
- Unlimited validation
- Full attribute control
- Field sensitive application help system
- Multiple virtual windows
- Fully automatic, collision proof overlay and restore
- Print to & scroll background windows
- Animated window "zoom"
- Move, grow, shrink, hide, or show any window
- Date & time arithmetic routines
- "Loop function" allows processing while awaiting input

...and much much more!

100 % MONEY BACK GUARANTEE

Better than a brochure. More informative than a testimonial. Our guarantee gives you the opportunity to see first hand why programmers who demand performance are using Vitamin C.

Find out for yourself why Vitamin C users are saying...

The best structured and documented C source package we have ever seen.

Thanks to Vitamin C, our projects are back on schedule.

I own them all, but I USE Vitamin C!

NEW! VCScreen NEW!

Our new interactive screen "painter" actually lets you draw your input screens. Define fields, text, boxes & borders. Move them around. Change their attributes. Then after everything is just the way you want it, the touch of a button generates C source code calls to Vitamin C routines OR generates parameter files that will dynamically load & build each screen at run time. Either way, your screen designs will be faster & more pleasing with VCScreen! Requires Vitamin C Library.

VCScreen \$99.95

Available for the IBM PC/XT/AT and compatibles.

For Orders Or More Information...

(214)245-6090

Creative Programming Consultants

Box 112097

Carrollton, Texas 75011-2097

Please add \$3 ground, \$6 air, \$15 overnight shipping per unit inside USA, \$25 per unit if outside USA. Texas residents add 6 1/8% sales tax. All funds must be in U.S. dollars drawn on a U.S. bank.

68K ASSEMBLER

(Continued from page 53)

address counter must be entered into the table. Because the same syntax analysis routines are used for both passes, some errors are reported during this pass.

Figure 1, page 55, is a data flow diagram for pass 1. The 68000 source code is read one line at a time and split into four parts by the routines in Parser. The definition module for Parser is given in Listing Three, page

76. LineParts is the only procedure that is exported by Parser, but several routines that are hidden in the implementation module do most of the work. (The implementation modules will be presented and explained in parts 2 and 3 of this series of articles.)

Parser passes any labels on to the SymbolTable module for entry in the symbol table; the opcodes (e.g., MOVE) go to the OperationCodes module where the machine code is extracted from a lookup table; and the operands (e.g., R0,(A2)) are sent,

via the BuildSymTable procedure in CodeGenerator, to the SyntaxAnalyzer module where their format is checked and their size determined. The definition modules for SymbolTable, CodeGenerator, and SyntaxAnalyzer are Listings Four, Five, and Six (pages 76 and 78).

If a label is present, *BuildSymTable* passes a value (most often the address count) to the SymbolTable module, where it is stored and referenced to its label. Although the SymbolTable module has four procedures for managing the symbol table, it is mostly the *FillSymTab* routine that finds work during pass 1. The definition module for ErrorX68 is shown as Listing 7, page 78 and is responsible for outputting error messages to the console and then returning to the main flow when the programmer acknowledges the error by pressing any key on the console keyboard.

ATRON BUGBUSTERS GREASE BORLAND LIGHTNING

"If I were starting a software company again, from scratch, Atron's AT PROBE™ would be among my very first investments. Without Atron's hardware-assisted, software debugging technology, the flash of Turbo Lightning™ would be a light-year away!"

Philippe Kahn, President, Borland

HOW BORLAND DOES SO MUCH, SO WELL, SO FAST

We asked Borland International president Philippe Kahn to share his secrets for rapidly taking a good idea and turning it into rock-solid reality. How does the Borland team do so much, so well, so fast?

He begins, "I remember when Atron was doing the June 24, 1985 *Wall Street Journal* chart of top-selling software in an ad." [Note: At that time, seven of the top ten software packages were created by Atron customers; it's now now nine out of ten.] "SideKick was number four, and I let Atron quote me in saying that there wouldn't have been a SideKick without Atron's hardware-assisted debuggers.

"You might say lightning has literally struck again. Turbo Lightning made number four on

SoftSel's Hotlist within weeks of its introduction! And again, I say we couldn't have done it without Atron debugging technology.

"Cleverly written code is, by definition tight, recursive, and terribly complex," he continues. "Without the ability to externally track the execution of this code, competent debugging becomes very nearly impossible."

Concludes Philippe, "And after Turbo Lightning was solid and reliable, Atron tuning software turned our Probes into performance analyzers. How do you think we greased our lightning?"

Philippe, along with a couple million or so of your satisfied customers, we say congratulations on yet another best-selling product. We can't wait to see what awesomely useful technology will come shooting out of Borland International next.

HOW BUGBUSTERS KEEP YOU FROM GETTING SLIMED

The AT PROBE is a circuit board that plugs into your PC/AT. It has an umbilical which plugs into the 80287 socket and monitors all 80286 activity.

Since AT PROBE can trace program execution in real time, and display the last 2048 memory cycles in symbolic or source-code form, you can easily answer the questions: "How did I get here?" and "What are those silly interrupts doing?"

It can solve spooky debugging problems. Like finding where your program overwrites memory or I/O-impossible with software debuggers.

You can even do source-level debugging in your favorite language, like C, Pascal or assembler. And after your application is debugged, the AT PROBE's performance measurement software can isolate performance bottlenecks.

Finally, the AT PROBE has its own 1-MByte of memory. Hidden and write-protected. How else could you develop that really large program, where the symbol table would otherwise take up most of memory.

LOOK AT IT THIS WAY.

History shows that non-Atron customers don't stand a very good chance of making the Top Ten list. Lightning really does have a way of striking twice!

The PC PROBE™ is \$1595 and the AT PROBE is \$2495. So call Atron today. You can be busting some really scary bugs tomorrow. And maybe, just like Borland, you can also bust some records.



20665 Fourth Street • Saratoga, CA 95070 • 408/741-5900
Copyright © 1985 by Atron Corp. PC PROBE™ and AT PROBE™ Atron, SideKick™ and Turbo Lightning™ Borland International, Inc. Adv. by TRBA, 408/258-2708.



Assembly—Pass 2

The major purpose of pass 2 is, of course, generation of machine code, which is written to an S-record file on disk. In addition, a formatted program listing is created, also on disk.

Before pass 2 actually starts, the

Specifications

- Two-pass assembler (no macros)
- Written entirely in high-level language
- Supports pseudo-ops
 - ORG & RORG
 - EQU
 - DC.B
 - DC.W
 - DC.L
 - DS
 - END

Creates formatted listing file

- Creates S-record file
- S0 header record
- S2 data/code records
- S8 trailer record

Outputs error messages to console

- Undefined opcode/pseudo-op
- Label defined twice
- Undefined label
- Operand inconsistent with op-code
- This addressing mode not allowed
- Phase error

Numeric constants/operations

- HEX
- Decimal
- Binary
- + / -

Table 1

SortSymTab procedure in the *SymbolTable* module sorts all identifiers into alphabetical order. This allows their values to be found more quickly during the code generation pass. Most of the steps taken in pass 1 are repeated essentially unchanged during pass 2. The data flow diagram for this pass is shown as Figure 2, below. Parser still performs the same task and passes labels, opcodes, and operands onto the same modules as before. During pass 2, however, it is the *GetObjectCode* procedure in *CodeGenerator* that works with the various procedures in the *SyntaxAnalyzer* module.

The two "busiest" routines in the whole process are *GetOperand* from *SyntaxAnalyzer* and a routine called *MergeModes* hidden within the implementation module of *CodeGenerator*. *GetOperand* determines the mode and values (if any) for all operands; *GetValue*, *GetSize*, and several other procedures help with the smaller jobs. *MergeModes* takes all the information from *OperationCodes*, *SymbolTable*, and *SyntaxAnalyzer* and combines it to produce hexadecimal machine code.

The *Listing* and *Srecord* modules use the machine code from *CodeGenerator* to create their files. *Listing* also gets the complete lines of source code from the *Parser* module to merge it with the object code for that line. The result is a formatted listing, with addresses, object code, source code, and page numbers. As an aid to debugging, the *ListSymTab* procedure in *SymbolTable* provides a sorted list of all identifiers, along with their values. The definition modules for *Listing* and *Srecord* are shown as Listings Eight and Nine, shown on page 78.

The Main Program

The main program for X68000 is shown as Listing Ten, page 78. From the above description, it should be obvious that there is not too much for the main module to do. Its tasks consist of inputting and formatting the 68000 source code file name; opening, resetting, and closing files; and providing the two *REPEAT* loops that control pass 1 and pass 2.

The most interesting aspect of these jobs is the command line interface. Because I hacked at C before I

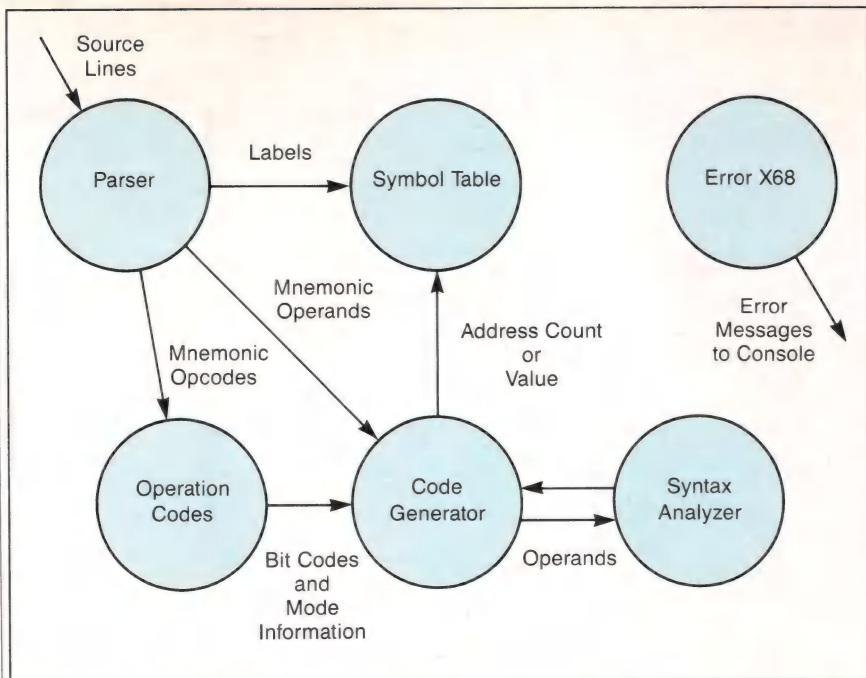


Figure 1: X68000—Data Flow for Pass 1

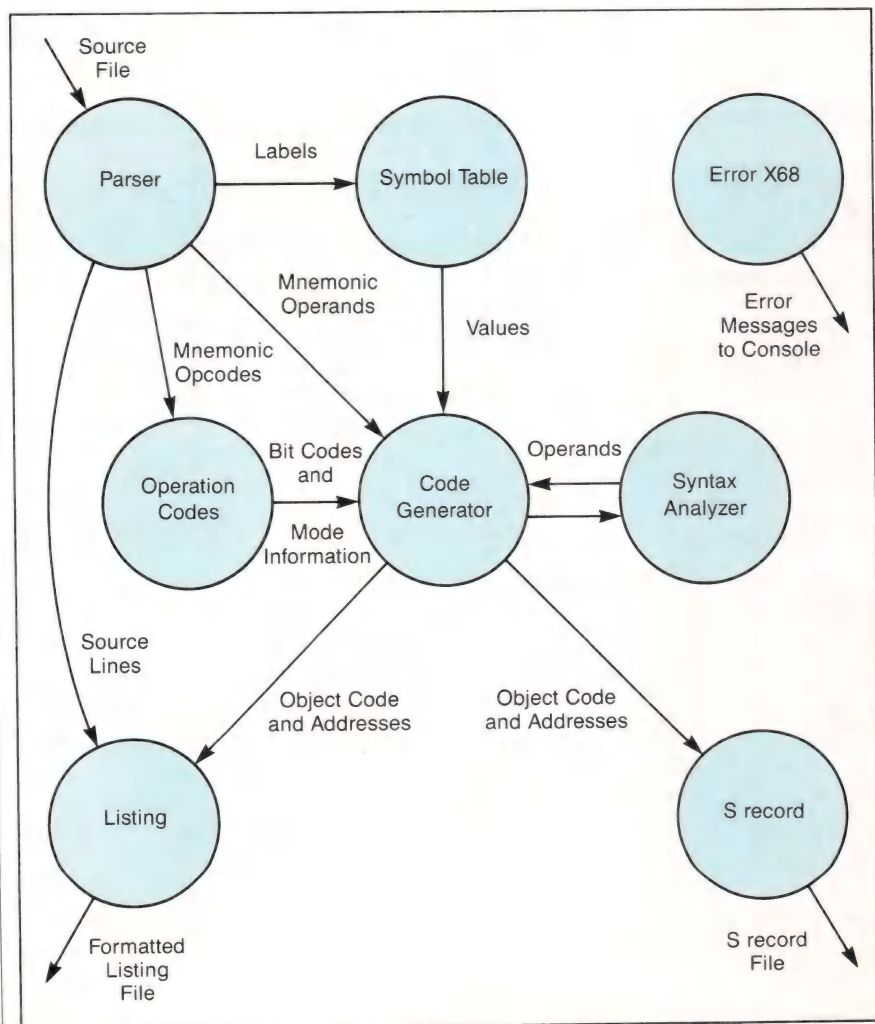


Figure 2: X68000—Data Flow for Pass 2

Make your PC or AT into a COMMUNICATING WORKSTATION for only \$85

Use ZAP, the Communications System for Technical Users COMPLETE Communications for PROGRAMMING and ENGINEERING

EMULATION of graphics and smart terminals is combined with the ability to TRANSFER files reliably, CAPTURE interactive sessions, and transmit MESSAGES while also being able to swap between your mini or mainframe session and your PC application. SUSPEND a line to run a PC application. Reconfigure features to fit the communications parameters and keyboard requirements of the host computer software. Complete technical documentation helps you understand and fit ZAP to your style.

HIGHLIGHTS OF ZAP:

- Emulate TEKtronix 4010/14 and DEC VT 100, 102, 52 including variable rows and columns, windows, full graphics, even half tones.
- Reliable *file transfer* to/from any mainframes and PCs including KERMIT and XMODEM protocols plus you get a full copy of KERMIT. Transfer *speeds* ranging from 50 to 38,400 BAUD. Session control include *printer dumps* and *save to disk*.
- **MACRO and Installation files** ("scripts") controllable by you.
- EMACS, EDT and VI "Script" files are included. ZAP is also used with other popular software including graphics products like DISPLA and SAS/GRAPH.
- **CONFIGURABLE** to communications, terminal features on the "other end"; 1, 2 stop bits; 5, 6, 7 or 8 data bits; parity of odd, even, none, mark and space; remap all keys including the numeric pad and standard keyboard, set any "virtual" screen size.
- Full PC/MSDOS access to run any command or program that will fit in your systems memory. ZAP takes less than 64K.
- 9 Comm ports are supported by ZAP. Plus full color in text and graphics make use of the IBM color, EGA cards, or Hercules Monochrome.

**ONLY
\$85**

Full refund if not satisfied
during first 30 days.

**Solution
Systems™**

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle no. 155 on reader service card.

The C Programmer's Assistant

C TOOLSET

UNIX-like Utilities for Managing C Source Code

No C Programmer should be without their assistant - C ToolSet from Solution Systems. The package consists of several utilities designed to help make C programming tasks easier.

C ToolSet (formerly C Helper) includes:

DIFF - Compares text files on a line-by-line basis or use CMP for byte-by-byte - indispensable for showing changes among versions of a program under development. So "intelligent" it stays in synch even when you add 100 lines.

GREP - Regular expression searches - ideal for finding a procedural call

or a variable definition amid a large number of header and source files.

FCHART - Traces the flow of control between the large modules of a program.

PP (C Beautifier) - Formats C program files so they are easier to read.

XREF (CCREF) - Cross references variables from a program.

**Source Code Included
Available For MS-DOS,
CP/M 86, CP/M 80 - \$95**

**Money Back Guarantee -
Try C ToolSet For 30 Days -
If Not Satisfied Get A
Full Refund.**

**TO ORDER CALL:
800-821-2492**

**Solution
Systems™**

335-R Washington St. Norwell, Mass. 02061 617-659-1571

68K ASSEMBLER

(Continued from page 55)

learned Modula-2, I sometimes miss some of the facilities provided by that language. (My article "Bit Manipulation in Modula-2," DDJ, November 1985, sprang from a similar hang-up.) As mentioned above, the module CmdLin2 provides facilities similar to C's via the ArgC and ArgV arguments. The declaration of ArgV lists it as an ADDRESS; in Modula-2, that makes it compatible with all pointer types. What CmdLin2 does internally is to create an array of pointers, with one pointing to each argument. ArgV points to that array. So, to use the ReadCmdLin procedure, I declare ArgV as:

```
POINTER TO ARRAY [1..n] OF POINTER  
TO STRING;
```

And each string becomes:

```
ArgV^[i]^.
```

Although this program has only one command argument (the file name), CmdLin2 was written as a general-purpose library routine. Incidentally, the 2 in the name is because the compiler comes with a library module called CmdLin, which uses a more conventional (for Modula-2) approach to the problem—you bring in the whole command line as a string and parse it into arguments yourself. It is interesting that the C approach results in a smaller module but does more work for you!

Availability

The following is available directly from the author for \$20 (U.S.):

1. A 25-page X68000 *User's Manual* that includes operating instructions for the program as well as a description and example of a method to use the assembler to link several modules.

2. An 8-inch CP/M SSSD disk or a 5¼-inch IPM PC disk with executable program as well as complete source code and several 68000 assembly-language examples. **DDJ**

(Listings begin on page 76.)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 5.

Brand New From Peter Norton A PROGRAMMER'S EDITOR

only
\$50

that's *lightning fast* with the *hot*
features programmers need

Direct from the
man who gave you
The Norton Utilities,
Inside the IBM PC,
and the *Peter Norton
Programmer's Guide*.

THE NORTON EDITOR



"This is the program-
mer's editor that I wished
I'd had when I wrote my
Norton Utilities. You can
*program your way to
glory* with *The Norton
Editor*."

Peter Norton



*Easily customized, and saved
Split-screen editing
A wonderful condensed/outline display
Great for assembler, Pascal and C*

Peter Norton, 2210 Wilshire Blvd., #186
Santa Monica, CA 90403, 213-826-8032
Visa, MasterCard and phone orders welcome

Circle no. 243 on reader service card.

Excellence

In your job, it depends on having the best tools available at your disposal. With such tools, your productivity increases and your work becomes easier.

Wisely, you keep a sharp eye open for products using the latest technology...Those truly representing the state of the art.

You have now located the source of advanced debugging technology for PC-DOS and CP/M-80. More powerful debugging software is not available anywhere...at any price. Yet the cost is affordable to even the smallest budget.

*DSD-80...Absolutely the most powerful
and easiest to use debugger for
CP/M-80. Full screen symbolic design
now includes a back tracing capability.
Only 125.00*

*DSD-86...New and innovative design
combines the most sophisticated user
interface with the most flexible display
to create a new generation of
debugging technology for the IBM PC.
Only 69.95*

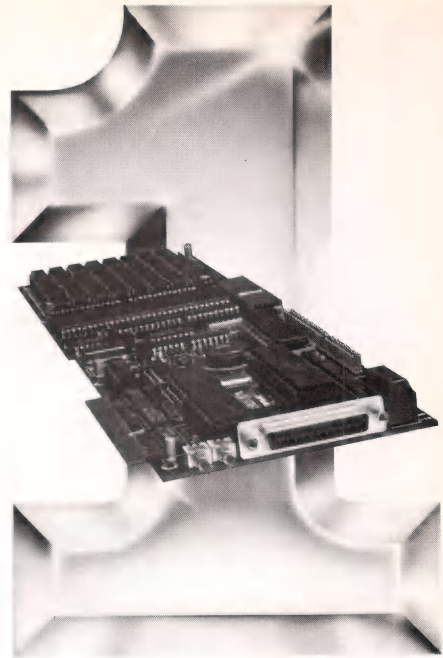


SoftAdvances

P.O. Box 49473 - Austin, Texas 78765 - (512) 478-4763

Visa & Mastercard Accepted. Please include 4.00 for shipping and handling.

Circle no. 83 on reader service card.



Number One in Performance

68010/68000 Coprocessor for IBM/AT/XT/PC-

8/10/12.5mz No Wait States

\$1295⁰⁰ Qty. 1

FEATURES

- 1-2 MB RAM (1MB Standard)
- 16K-64K EPROM
- 2-8 Serial Ports
Async/Sync/Bisync Communications
- Battery-backed Real Time Clock
- Battery-backed 2K-8K RAM
- 2 Parallel Ports
- 68881 Math Coprocessor
- Memory-mapped Dual-port BUS
- 3-9 Users Per Board (3 Standard)
- Up To 16 Boards Per AT/XT/PC
- Can Operate As Standalone Processor

SOFTWARE

- OS9 (Powerful UNIX-like Multi-user OS)
- CPM/68K
- Software selectable OS including concurrent PC DOS/OS-9 or CPM/68K operation
- Support Module for IBM Graphics
- High-speed Local/Global Disk Caching
- Basic, Pascal, Fortran, C, and COBOL

IBM is a registered trademark of International Business Machines Corporation. OS/2 is a registered trademark of Micro Channel Systems Corp. CPM is a registered trademark of Digital Research Corp. Microsoft, MS-DOS are registered trademarks of Microsoft. UNIX is a registered trademark of AT&T.



West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
Distributor: Telemarketing Services, Inc.
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

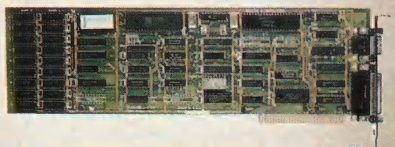
Circle no. 173 on reader service card.

THE WORLD'S FASTEST Z-80 COMPUTER IS NOW A PC



EARTH COMPUTER's **TURBOSLAVE-PC™** is the world's fastest Z-80 Coprocessor. Running at 8MHz, it was designed to permit operation of thousands of CP/M application programs on your IBM-PC, XT, AT™, or compatible computer system.

The **TURBOSLAVE-PC** supports the TurboDOS™ multi-user operating system which allows up to 16 users on your PC. It is the only IBM-PC/Z-80 system that is MP/M™ compatible and allows **TRUE** multi-user, multi-process operations, including full record locking and security.



Discover a whole new world of high-speed (8MHz) single and multi-user applications for your personal computer. Discover the **TURBOSLAVE-PC**... the world's fastest Z-80 Coprocessor, with such outstanding features as:

- 128K RAM with parity
- 2 Serial ports
- On-board Counter Timer
- S.L.R. Z-80 assembler included

To order your **TURBOSLAVE-PC**, call or write to:



EARTH COMPUTERS

P.O. Box 8067, Fountain Valley, CA 92728
TELEX: 910 997 6120 EARTH FV

(714) 964-5784

Ask about **EARTH COMPUTERS'** other fine PC and S-100 compatible products.

IBM-PC, XT, AT are trademarks of International Business Machines, Inc.; CP/M and MP/M are trademarks of Digital Research; TurboDOS is a trademark of Software 2000; **TURBOSLAVE-PC** is a trademark of Earth Computers

C CHEST

LISTING ONE (Text begins on page 18.)

Listing 1 -- redir.c

```

1 redir( cmdline )
2 register char *cmdline;
3 {
4     /*      Handles redirection. The command line will be null
5     *      terminated wherever the first < or > is found.
6     *      Returns 1 if any redirection happened, 0 if none.
7     */
8
9     register int    inquote = 0;
10    int            rval    = 0;
11    int            input;
12    int            append;
13    int            erralso;
14    char           *fname;
15
16    while( *cmdline )
17    {
18        /* Skip to a < or >. Brackets in quoted strings or
19        * preceded by a \ are ignored. When the loop terminates
20        * cmdline will be pointing to end of string or the
21        * angle bracket.
22        */
23
24
25        for( ; *cmdline && (inquote || !ISREDIR(*cmdline)); cmdline++ )
26        {
27            if( *cmdline == '\\' && *(cmdline+1) )
28                cmdline++;
29
30            else if( ISQUOTE(*cmdline) )
31                inquote = ~inquote;
32        }
33
34        if( !*cmdline )
35            break;
36
37
38        /* If we get here then we're processing a < or >
39        * Parse the command, and strip out the file name.
40        */
41
42
43        rval = 1;
44        input = (*cmdline == '<');          /* < or << */
45        *cmdline++ = '\0';
46
47        if( append = ISREDIR(*cmdline) )    /* << or >> */
48            cmdline++;
49
50        if( erralso = (*cmdline == '&') )    /* >& or >>& */
51            cmdline++;
52
53        SKIPWHITE(cmdline);                  /* skip to file */
54        fname = cmdline;                     /* name. */
55
56        while( *cmdline && !isspace(*cmdline) && !ISREDIR(*cmdline) )
57            cmdline++;
58
59        if( *cmdline )
60            *cmdline++ = '\0';
61
62
63        /* Now actually do the redirection
64        */
65
66
67        if( input )
68            freopen( fname, "r", stdin );
69        else
70        {
71            freopen( fname, append ? "a" : "w", stdout );
72            if( erralso )
73                dup2( 1, 2 );
74        }
75
76        return rval ;
77    }
78 }
79
80 /*-----*/
81
82 unredir()
83 {
84     freopen( "/dev/con", "r", stdin );
85     freopen( "/dev/con", "w", stdout );
86     dup2( 1, 2 );
87 }
88

```

End Listing One

LISTING TWO

Listing 2 -- switch.c

```

1 #include <stdio.h>
2 #include <dos.h>
3
4 /* SWITCHAR.C:      Read or set the switch character depending
5 *                  on whether one is present on the command
6 *                  line.
7 *
8 * Author: Anthony LiCausi
9 *
10 * Modified somewhat by Allen Holub.
11 */
12
13 int      switchar( c )
14 {
15     /* If c == 0, return the current switch character, else
16     * change the switch character to c and return the old
17     * switch character. The routine is mildly recursive.
18     */
19
20     union REGS      regs;
21     register int     rval = 0;
22
23     if( c )
24         rval = switchar(0);
25
26     regs.x.dx = c;
27     regs.x.ax = c ? 0x3701 : 0x3700 ;
28
29     intdos( &regs, &regs );
30
31     return rval ? rval : regs.h.dl ;
32 }
33
34 /*-----*/
35
36 main( argc, argv )
37 char  **argv;
38 {
39     argv++;
40
41     if( argc > 1 )
42         printf("Changing switch character from <%c> to <%c>\n",
43                switchar(**argv), **argv);
44     else
45         printf("Switch character is <%c>\n", switchar(0) );
46
47     exit(0);
48 }
49

```

End Listing Two

LISTING THREE

Listing 3 -- touch.c

```

1 #include <stdio.h>
2 #include <fcntl.h>
3
4 /* TOUCH.C -      Touches a files date and time so as to make
5 *                  the file current. Usage is:
6 *                  touch file [file ...]
7 *
8 * Author: Michael Yam
9 * Public Domain (P) October 1985
10 *
11 * Modified so that a file is created if it doesn't exist - AH
12 */
13
14 /*-----*/
15
16 #ifdef LATTICE
17
18 #define EXISTS    O_RAW|O_RDWR
19 #define CREATE    O_CREAT|EXISTS
20
21 #else
22
23 #include <types.h>
24 #include <stat.h>
25 #define EXISTS    O_BINARY|O_RDWR
26 #define CREATE    O_CREAT|EXISTS, S_IREAD|S_IWRITE
27
28 #endif
29
30 /*-----*/
31
32 main( argc, argv )
33 char  **argv;
34 {

```

(continued on next page)



Number One In Performance

Hard Disk Intelligent VCR Backup for AT/XT/PC

FEATURES

- High speed microprocessor controlled backup (68000)
- Two channel interface
- Built in LAN channel
- Software control of most VCR functions including Fast Forward, Rewind, and auto backup using VCR timer capabilities
- Economical VHS or Beta formats



West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
 East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
 Distributor: Telemarketing Services, Inc.
 1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle no. 174 on reader service card.

LISTING THREE (Listing continued, text begins on page 18.)

```

35     int    buffer = '\0';
36     int    err_lvl = 0;
37     int    file;
38
39     for( ++argv; --argc > 0; ++argv )
40     {
41         if( (file = open(*argv, EXISTS)) != -1)
42         {
43             /* File exists */
44
45             read ( file, &buffer, 1 ); /* read a char */
46             lseek( file, 0L, 0 ); /* go back to start */
47             write( file, &buffer, 1 ); /* write the same char */
48             close( file );
49         }
50         else if( (file = open(*argv, CREATE)) != -1)
51         {
52             /* Created new file. Don't modify it so that
53              * it will remain zero length.
54              */
55
56             close( file );
57         }
58         else
59         {
60             /* File doesn't exist and can't be created */
61
62             err_lvl = 1;
63             fprintf(stderr, "Can't touch %s\n", *argv );
64         }
65     }
66     exit( err_lvl );
67 }

```

End Listings

Time and Money.

We've just done something we know you'll like. We've made the SemiDisk far more affordable than ever before. With price cuts over 25% for most of our product line. Even our new 2 megabyte units are included.

It's Expandable

SemiDisk Systems builds fast disk emulators for more microcomputers than anyone else. S-100, IBM-PC, Epson QX-10, TRS-80 Models II, 12, and 16. You can start with as little as 512K bytes, and later upgrade to 2 megabytes per board...at your own pace, as your needs expand. Up to 8 megabytes per computer, using only four bus slots, max! Software drivers are available for CP/M 80, MS-DOS, ZDOS, TurboDOS, VALDOCS 2, and Cromix. SemiDisk turns good computers into **great** computers.

SEMIDISK

SemiDisk Systems, Inc., P.O. Box GG, Beaverton, Oregon 97075 503-642-3100

Call 503-646-5510 for CBBS/NW, 503-775-4858 for CBBS/PCS, and 503-649-8327 for CBBS/Aloha. All SemiDisk equipped computer bulletin boards (300/1200 baud) SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Circle no. 85 on reader service card.

Battery Backup, Too

At 0.7 amps per 2 megabytes, SemiDisk consumes far less power than the competition. And you don't have to worry if the lights go out. The battery backup option gives you 5-10 hours of data protection during a blackout. Nobody else has this important feature. Why risk valuable data?

The Best News

	<u>512K</u>	<u>1Mbyte</u>	<u>2Mbyte</u>
SemiDisk I, S-100	\$695	\$1395	
SemiDisk II, S-100	\$995		\$1995
IBM PC, XT, AT	\$595		\$1795
QX-10	\$595		\$1795
TRS-80 II, 12, 16	\$695		\$1795
Battery Backup Unit	\$150	\$150	\$150

Someday you'll get a SemiDisk.
Until then, you'll just have to....wait.



AVAILABLE BACK ISSUES

1982	1983	1984	1985
No. 68—June	No. 77—March	No. 87—Jan.	No. 99—Jan.
No. 69—July	No. 78—April	No. 88—Feb.	No. 101—March
No. 70—Aug.	No. 79—May	No. 90—April	No. 102—April
No. 71—Sept.	No. 80—June	No. 91—May	No. 104—June
No. 72—Oct.	No. 81—July	No. 92—June	No. 105—July
No. 73—Nov.	No. 82—Aug.	No. 93—July	No. 106—Aug.
	No. 83—Sept.	No. 95—Sept.	No. 107—Sept.
	No. 84—Oct.	No. 96—Oct.	No. 108—Oct.
	No. 85—Nov.	No. 97—Nov.	No. 109—Nov.
	No. 86—Dec.		No. 110—Dec.

TO ORDER: send \$5 for 1 issue, \$4.50 each for 2-5, \$4 each for 6 or more
to: Dr. Dobb's Journal, 501 Galveston Drive, Redwood City, CA 94063

Name _____

Address _____

City _____

State _____

Zip _____

3114G

C68 & C68/020 C COMPILERS for MC680X0

NOW AVAILABLE
ON IBM PC

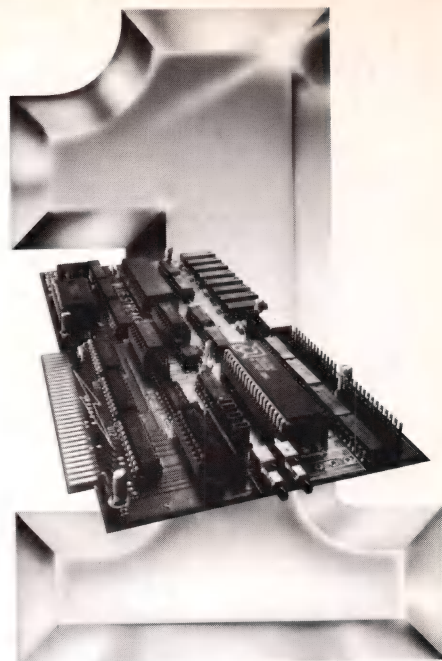
- ▶ Produce highly optimized code
- ▶ Complete development environment: Assembler, Linking and Downline Loaders, Runtime Libraries
- ▶ Available for Motorola, DEC, and Alcyon host computers
- ▶ The #1 choice for compact, fast MC680X0 code
- ▶ \$1495 for C68 (Motorola host)
\$2295 for C68/020 (Motorola host)

Alcyon
CORPORATION

5010 Shoreham Place
San Diego, CA 92122
(619) 587-1155 TELEX 5106004947

DEC is a Trademark of Digital Equipment Corporation.

Circle no. 221 on reader service card.



**Number One
in Performance**

**Z80H
BLUESTREAK™**

**IBM/AT/XT/PC- 8mz
No Wait States**

FEATURES

- 64K-256K RAM
- 2K-8K EPROM/Static Ram
- 2 Serial Ports
- Async/Sync/Bisync Communications
- Real Time Clock
- Memory-mapped Dual-port BUS
- On-board/Remote Reset NMI capability
- Up To 32 Boards Per AT/XT/PC
- Can Operate As Standalone Processor
- Less Than Full Size Board
(will fit other compatibles.)

SOFTWARE

- ZP/M™ CP/M Emulation Software
(Supports Most CP/M Software)
- Multiuser Capability If Used As A
Slave Processor

IBM is a registered trademark of International Business Machines.
CPM 80 is a registered trademark of Digital Research Corp.



West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
Distributor: Telemarketing Services, Inc.
1897 Garden Ave., Eugene, OR 97403, 503/345-7395

Circle no. 175 on reader service card.

INFERENCE ENGINE

LISTING ONE (Text begins on page 24.)

elijah-brown father-of robert-brown-sr
 robert-brown-sr father-of robert-brown-jr
 john-mccollister father-of lavenia-mccollister
 isam-mccollister father-of john-mccollister
 mr-holt father-of bettie-holt
 elias-presley father-of margret-presley
 robert-brown-jr father-of robert-brown-iii
 paul-h-sewall father-of virginia-sewall
 paul-h-sewall father-of paul-sewall-jr
 paul-sewall-jr father-of peter-sewall
 paul-sewall-jr father-of dee-dee-sewall
 paul-sewall-jr father-of mark-sewall
 paul-sewall-jr father-of paul-sewall-iii
 robert-brown-jr father-of kenneth-brown
 robert-brown-sr father-of amos-trice-brown
 robert-brown-sr father-of james-elro-brown
 clarence-bailey-compton father-of elanor-compton
 george-washington-compton father-of clarence-bailey-compton
 samuel-compton father-of george-washington-compton
 henry-sewall father-of paul-h-sewall
 dr-james-d-nelson father-of rachel-nelson
 robert-sewall father-of henry-sewall
 paul-hebert father-of evelina-hebert
 harry-breedon father-of X if
 dorothy-wallace mother-of X
 berke-breedon father-of harry-breedon
 john-wallace father-of dorothy-wallace
 robert-brown-iii father-of krystl-raquelle-brown
 paul-sewall-jr father-of robert-sewall-ii
 danny-crider father-of amy-crider
 bill-skirvin father-of marty-skirvin
 bill-skirvin father-of rodney-skirvin
 tommy-breedon father-of thomas-andrew-breedon
 tommy-breedon father-of suzanne-breedon
 john-alsop father-of joy-alsop
 lester-stevens father-of geneva-stevens
 strawder-breedon father-of daren-breedon
 daren-breedon father-of shaun-breedon
 strawder-breedon father-of deidra-breedon
 robert-bishop father-of krystal-bishop
 robert-bishop father-of tiffany-bishop
 robert-brown-sr father-of opal-brown
 strawder-breedon father-of deva-breedon
 strawder-breedon father-of stephanie-breedon
 johnny-wilson father-of jonathan-wilson
 lester-stevens father-of geneva-stevens
 bettie-holt mother-of lavenia-mccollister
 miss-hornsbey mother-of robert-brown-sr
 elizabeth-arthur mother-of john-mccollister
 margret-e-presley mother-of betty-holt
 virginia-sewall mother-of robert-brown-iii
 elanor-compton mother-of virginia-sewall
 lillian-givens mother-of X if
 paul-sewall-jr father-of X
 virginia-sewall mother-of kenneth-brown
 lavenia-mccollister mother-of amos-trice-brown
 lavenia-mccollister mother-of james-elro-brown
 sarah-virginia-sanford mother-of elanor-compton
 mary-eliza-sanford mother-of clarence-bailey-compton
 rachel-nelson mother-of paul-h-sewall
 jane-kirk mother-of rachel-nelson
 evelina-hebert mother-of henry-sewall
 eugina-hamilton mother-of evelina-hebert
 dorothy-wallace mother-of darlene-breedon
 dorothy-wallace mother-of willena-breedon
 dorothy-wallace mother-of karen-breedon
 dorothy-wallace mother-of bonnie-breedon
 dorothy-wallace mother-of tommy-breedon
 dorothy-wallace mother-of dorothy-breedon
 dorothy-wallace mother-of brenda-breedon
 dorothy-wallace mother-of betty-breedon
 dorothy-wallace mother-of strawder-breedon
 flo-marsh mother-of harry-breedon
 christine-xxx mother-of dorothy-wallace
 darlene-breedon mother-of krystl-raquelle-brown
 dorothy-breedon mother-of amy-crider
 brenda-breedon mother-of X if
 bill-skirvin father-of X
 joy-alsop mother-of suzanne-breedon
 joy-alsop mother-of thomas-andrew-breedon
 pauline-davis mother-of joy-alsop
 myrtle-jackson mother-of geneva-stevens
 tammy-xxx mother-of shaun-breedon
 deidra-breedon mother-of krystal-bishop
 deidra-breedon mother-of tiffany-bishop
 lavenia-mccollister mother-of opal-brown
 lavenia-mccollister mother-of robert-brown-jr
 elanor-compton mother-of paul-sewall-jr
 karen-breedon mother-of jonathan-wilson
 geneva-stevens mother-of X if
 strawder-breedon father-of X
 myrtle-jackson mother-of geneva-stevens
 male (X) if
 X father-of Y
 male (john-viscar)
 male (shaun-breedon)
 female (X) if
 X mother-of Y
 female (willena-breedon)
 female (bonnie-breedon)
 female (karen-breedon)
 female (betty-breedon)
 female (judy-tracey)
 X wife Y if

X mother-of Z and
 Y father-of Z
 X husband Y if
 Y wife X
 X parent-of Y if
 X father-of Y
 X parent-of Y if
 X mother-of Y
 X wife-of Y if
 X wife Y
 judy-tracy wife-of tommy-breedon
 X husband-of Y if
 X husband Y
 john-viscar husband-of pauline-davis
 X child-of Y if
 Y parent-of X
 X descendant-of Y if
 X child-of Y
 X descendant-of Y if
 Z child-of Y and
 X descendant-of Z
 X ancestor-of Y if
 X parent-of Y
 X ancestor-of Y if
 Z parent-of Y and
 X ancestor-of Z
 lavenia-mccollister mother-of robert-brown-jr
 X sibling-of Y if
 Z mother-of X and
 Z mother-of Y and
 X father-of X and
 X father-of Y and
 Not (X EQ Y)
 X half-sibling-of Y if
 Z mother-of X and
 Z mother-of Y and
 Not (X sibling-of Y) and
 Not (X EQ Y)
 X half-sibling-of Y if
 Z father-of X and
 Z father-of Y and
 Not (X sibling-of Y) and
 Not (X EQ Y)
 X aunt-or-uncle-of Y if
 Z parent-of Y and
 X sibling-of Z
 X cousin-of Y if
 Z aunt-or-uncle-of X and
 Y child-of Z

End Listing One

LISTING TWO

which (X X father-of robert-brown-iii)
 Answer is robert-brown-jr
 No (more) answers

which (X robert-brown-iii father-of X)
 Answer is krystl-raquelle-brown
 No (more) answers

which (X X parent-of krystl-raquelle-brown)
 Answer is robert-brown-iii
 Answer is darlene-breedon
 No (more) answers

which (X X descendant-of robert-brown-sr)
 Answer is robert-brown-jr
 Answer is amos-trice-brown
 Answer is james-elro-brown
 Answer is opal-brown
 Answer is robert-brown-iii
 Answer is kenneth-brown
 Answer is krystl-raquelle-brown
 No (more) answers

which (X X aunt-or-uncle-of robert-brown-iii)
 Answer is amos-trice-brown
 Answer is james-elro-brown
 Answer is opal-brown
 Answer is paul-sewall-jr
 No (more) answers

which (X X aunt-or-uncle-of krystl-raquelle-brown)
 Answer is kenneth-brown
 Answer is willena-breedon
 Answer is karen-breedon
 Answer is bonnie-breedon
 Answer is tommy-breedon
 Answer is dorothy-breedon
 Answer is brenda-breedon
 Answer is betty-breedon
 Answer is strawder-breedon
 No (more) answers

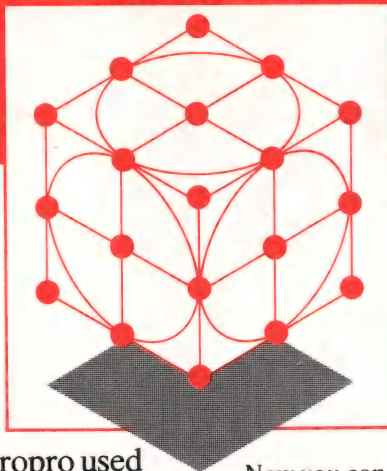
which (X robert-brown-iii cousin-of X)
 Answer is peter-sewall
 Answer is dee-dee-sewall
 Answer is mark-sewall
 Answer is paul-sewall-iii

(continued on page 66)

New from Logitech.

MODULA-2/86 VERSION 2.0

Professional Modula-2 for \$89.



Now the same powerful tools Micropro used to develop its latest word processing system is available to you at a new \$89.00 price.

Building Blocks for Tomorrow's Technology

Universities are switching to LOGITECH MODULA-2. Innovative programmers now develop applications and products with LOGITECH MODULA-2. The most productive teams at major companies depend on LOGITECH MODULA-2.

Now you can create your professional software development system using the proven technical sophistication of LOGITECH MODULA-2/86.

Systems to Fit Your Needs.

Base Language System

- Compiler and Linker
- Module Library

\$89

Base Language System/8087

- Inline 8087 code.

\$129

Base Language System/512K

- Full 8087 support.
- Uses RAM to increase speed by 40 to 50 percent.
- 80186 and 80286 support.

\$189

Run-Time Debugger

- Monitors the execution of a program with user-defined breakpoints or by stepping through the program.
- Symbolically displays the source code, data, procedure call chain, and raw memory.

\$69

MODULA-2 Editor

- Fast on-line Modula-2 syntax check.
- Can run compiler and linker from the editor.
- User definable templates for Modula-2 syntax constructs.

\$59

Utilities Package

- Decoders: Disassemble link and load files.
- Version: Administrate different versions of one program.
- Post-Mortem Debugger: Debugs a program after abnormal termination.
- Cross Reference: Produces a cross-reference listing of a Modula-2 program.

\$49

Sources

- Sources to customize your system.
- Run-Time System sources.
- Some library module sources.

\$179

Not Copy Protected

INTRODUCTORY OFFER

Through the end of April you get the new MODULA-2 Editor for free with any purchase of the Base Language System.

To place an order call our special toll free number:

800-231-7717

In California:

800-552-8885

YES, I want to create my professional software development system. Please send me the following building blocks:

- ☐ BLS \$89 ☐ BLS/8087 \$129 ☐ BLS/512K \$189
☐ RTD \$69* ☐ EDITOR \$59*
☐ UTILITIES \$49* ☐ SOURCES \$179*

*\$10 less with the purchase of any Base Language System. Please add \$5 for shipping and handling.

- ☐ VISA ☐ MASTERCARD ☐ CHECK ENCLOSED

CARD NUMBER _____ EXPIRATION DATE _____

SIGNATURE _____

NAME _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____ PHONE (____) _____



LOGITECH

LOGITECH, Inc.

805 Veterans Blvd., Redwood City, CA 94063, USA

Telephone: (415) 365-9852

LOGITECH SA

Box 32, CH-1143 Apples, Switzerland

Telephone: 41 (21) 774545

A single company has revolutionized the business of language, the language of business, and created the exciting new world of electronic reference works—Borland!

Which isn't bad for a 3-year-old. Turbo Pascal®, our first product, now has more than half a million users, and has become a world-wide standard. And that was just the beginning.

Since then, the Turbo Pascal family has grown to a family of 9, and today we're announcing our *second language*, Turbo Prolog®, the natural language of Artificial Intelligence.

We've also introduced amazing business productivity tools like SideKick®, Traveling SideKick®, Reflex, The Analyst®, and SuperKey®.

We broke new ground in 1985 with Turbo Lightning.™ It includes the Random House® dictionary and thesaurus. Turbo Lightning is the forerunner of a complete electronic reference library, newly joined by the Word Wizard,™ which solves the unsolvable twists, and boggles and challenges your mind. Word Wizard



also includes Turbo Pascal source code so you can figure out how the Turbo Lightning access system works.

And here is a brief synopsis of current offerings from the Borland library of history-making software...

Turbo Pascal® 3.0

The fastest Pascal compiler, plus an integrated programming environment. Includes a free MicroCalc™ spreadsheet, and 1,200 lines of annotated source code, ready to compile and run. Minimum memory: 128K.

Turbo Tutor®

Takes you from basic right through advanced programming concepts and techniques. Includes 300-page tutorial and source code for every example used in the reference manual. Minimum memory: 128K.

Turbo Graphix Toolbox™

Lets you create high-resolution graphics. Includes tools for complex business graphics, easy windowing, and storing screen images to memory. Complete with source code on disk, ready to compile. Minimum memory: 128K.

Turbo Database Toolbox™

Perfect complement to Turbo Pascal. Contains complete library of Pascal procedures that allows you to search and sort data and build powerful database applications. Minimum memory: 128K.

Turbo Editor Toolbox™ NEW!

It's all you need to build your own text editor or word processor. Provides all the routines—you decide which features you want. Source code included. Also includes the MicroStar™ text editor with pull-down menus and windowing. Interfaces directly with Turbo Lightning to let you spell-check your MicroStar files. Minimum memory: 192K.

Turbo GameWorks™ NEW!

Reveals the secrets and strategies of game theory. Includes source code so you can write your own games. Gives you ready-to-play Chess, Bridge, and Go-Moku, an ancient Japanese game that will provide hours of fascinating diversion. Even if you don't want to write your own games, it's a terrific value. Minimum memory: 192K.

NEW! NEW! NEW!
Turbo Prolog™
The natural language of Artificial Intelligence, Turbo Prolog is our second language and the latest product in the Borland software library. Turbo Prolog is a fifth-generation language, and probably the most powerful programming language ever conceived. Includes a 200-page reference manual and free GeoBase,™ a natural query language database with commented source code on disk, ready to compile. It's all you need to know about Artificial Intelligence at a Humanly Intelligent price. Minimum memory: 384K.

NEW!
Word Wizard™
Intriguing new addition to the Turbo Lightning Library.™ Solves unsolvable crosswords, and challenges your word skills and ability to break codes and ciphers. Scrambles, twists, turns, and boggles your mind. Includes Turbo Pascal source code and all the technical information you'll need to figure out the "nuts and bolts" of the Turbo Lightning access system. Minimum memory: 256K.

SideKick®
Powerful desktop management program. #1 best-seller for the IBM® PC. Includes notepad, calculator, appointment scheduler, telephone directory and autodialer, and ASCII table. RAM-resident, it's always there to help, and stays in the background while you run other programs. One keystroke activates it. Minimum memory: 128K.

Reflex, The Analyst™
Unique, easy-to-use database management and analysis. Shows your spreadsheet data from 1-2-3® dBase®, and others in five graphic forms—including bar charts, pie charts, scatter plots, line graphs, and stacked bar charts. Answers What If? questions. Minimum memory: 384K.

Turbo Lightning™
An electronic reference library which includes the 80,000-word Random House Concise Dictionary and the 50,000-word Random House Thesaurus. Checks your spelling as you type. Gives you instant synonyms. Leads the revolutionary way in electronic publishing. Minimum memory: 256K.

SuperKey®
Amazing keyboard enhancer for your IBM PC. With easy-to-write macros that can turn 1,000 keystrokes into 1. Also includes powerful encryption technology that keeps confidential files confidential; locks your keyboard with secret password protection. (Because of encryption technology, SuperKey is under a US Government export ban.) Minimum memory: 128K.

Traveling SideKick™ NEW!
BinderWare® that includes an organizer, a binder, a software program, and a report generator that picks your SideKick's electronic brain, then prints out your appointments, daily/weekly/monthly/yearly calendar, phone lists, mailing labels, or whatever else you need when you're away from your desk. It's the smart new way to take your computer with you without taking your computer with you. Minimum memory: 256K.

Special Prices!
You can save even more through September 1, 1986:
• Turbo Jumbo Pack—six Turbo products for only \$245.00: Turbo Pascal, Turbo Tutor, Turbo Graphix, Turbo Database Toolbox, Turbo Editor Toolbox, and Turbo GameWorks.
• SideKick and Traveling SideKick for only \$125.00.
• SideKick and Traveling SideKick and SuperKey, for only \$175.00.
• Turbo Lightning and Word Wizard for only \$149.95.

Turbo Prolog 1.0

Programming System features

Compiler

Incremental compiler generating native in-line code and linkable object modules. The linking format is compatible with the PC-DOS linker. Large memory model support. Compiles over 2500 lines per minute on standard IBM PC.

Interactive editor

System includes a powerful full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code. At run-time, Turbo Prolog programs can call the editor and view the running program's source code.

Type system

A flexible object-oriented type system is supported.

Windowing support

System supports both graphic and text windows.

Input/Output

Full I/O facilities including formatted I/O, streams, and random access files.

Numeric Ranges

Integers: -32768 to 32767;
Reals: 1E-307 to 1E+308

Turbo Prolog and GeoBase are trademarks and Turbo Pascal is a registered trademark of Borland International. IBM and AT are registered trademarks of International Business Machines Corp. MS-DOS is a registered trademark of Microsoft Corp.

PRIORITY

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 200 SANTA CRUZ, CA

POSTAGE WILL BE PAID BY ADDRESSEE

BORLAND
INTERNATIONAL

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CALIFORNIA 95066

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

Yes, Rush Me Turbo Prolog Today!

Send me _____ copies at:

\$99.⁹⁵

Introductory offer, good through October 1986
Price includes shipping to all US cities

NOT COPY-PROTECTED
***60-DAY MONEY-BACK GUARANTEE**

To order by phone or for the dealer nearest you,
call (800) 255-8008;
in CA call (800) 742-1133.

Name: _____

Shipping Address: _____

City: _____

State: _____

Telephone: _____

Subtotal: _____

Outside USA add \$10 per copy:

CA and MA res. add applicable sales tax: _____

Amount enclosed: _____

Payment: _____

VISA

MC

Bank Draft

Check

Credit card expiration date: ____/____/____

Card # _____

Minimum System Requirements:

Computer: IBM PC, XT, AT, Portable and true IBM compatibles
Operating system: PC-DOS 2.0 or later, MS-DOS 2.0 or later
Minimum system memory: 384K RAM

CODs and purchase orders WILL NOT be accepted by Borland. Outside USA make payment by bank draft payable in US dollars drawn on a US bank.

*YES, within 60 days of purchase, should you find that this product does not perform in accordance with our claims and representations, just call our customer service department and we will gladly arrange for a refund.

Other Borland Products include Turbo Pascal, Turbo Tutor, Turbo Lightning, Turbo Database Toolbox, Turbo Graphics Toolbox, Turbo Editor, Turbo GameWorks, SuperKey, SideKick, SideKick, The Macintosh Office Manager, Rellex, The Analyst and Traveling SideKick—all of which are registered trademarks or trademarks of Borland International or Borland Analytics, Inc.

The astonishing fact is that Borland's Turbo Prolog gives you a \$4 billion value for only \$99.95

More than four billion US dollars have already gone into the Japanese efforts to develop a 5th-generation super-computer.

Prolog is probably the most powerful computer programming language ever conceived—which is perhaps why our Japanese friends chose Prolog to be their super-computer's implementation language.

Without being rude about their efforts, we are pleased and proud to announce that they're not even close to achieving the performance that Turbo Prolog now brings to your standard IBM® PC.



Turbo Prolog is several billion dollars cheaper and several times faster than the swiftest Japanese experimental models. For only \$99.95, Turbo Prolog makes your desktop IBM PC outperform the Japanese 5th-generation computers—and if you're using an IBM AT® or equivalent machine, you're so far ahead of the Japanese that it's time to say *sayonara*.

Turbo Prolog is to Prolog what Turbo Pascal® is to Pascal!

Our Turbo Pascal astonished everyone who thought of Pascal as "just another language." We changed all that. And now Turbo Pascal is the *de facto* worldwide standard, with hundreds of thousands of enthusiasts and users in universities, research centers, schools, and with professional programmers, students, and hobbyists.

You can expect at least the *same impact* from Turbo Prolog, because while Turbo Prolog is the most revolutionary and natural programming language, it is also a complete programming environment—just like Turbo Pascal.

Turbo Prolog radically alters and dramatically improves the brave new world of artificial intelligence—and invites you into that fascinating universe for a *humanly intelligent* \$99.95.

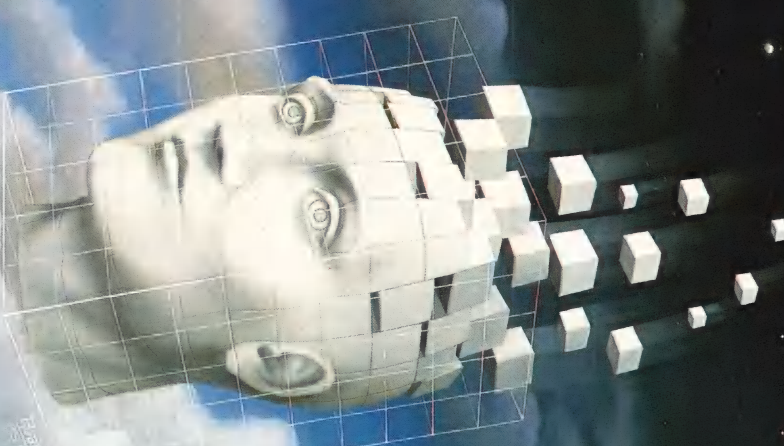
So how does Turbo Prolog work?

This quick little example shows you how Turbo Prolog functions. For the sake of example, imagine the following question:

Q: What does this man do on Sundays?

First you would need to tell your Turbo Prolog what you already know, and what it needs to know to answer the question—so you type in the information:

- A: He has poor eyesight.
B: He also cannot distinguish between the colors white and green.



C: His favorite day as a schoolboy was Flag Day.
D: He works on Sundays.

Then, assuming that you have taught Turbo Prolog the right set of rules, your Turbo Prolog deduces that if "A" is true and "B" is true and "C" is true and "D" is true, then the logical answer is:

A: On Sundays, this man probably works as a *National Football League* referee!

Of course, in case you're a referee and resent Turbo Prolog's conclusion, you could teach it to come to a completely different conclusion. All you need to do is change the information or change the rules—which an NFL referee would never do! Turbo Prolog's deductive powers are more than elementary, my dear Watson.

You get the complete Turbo Prolog programming system—including GeoBase™—for only \$99.95

Includes:

- The lightning-fast Turbo Prolog incremental compiler and the interactive Turbo Prolog Editor.
- The 200-page Turbo Prolog Reference Manual which includes the step-by-step Tutorial.
- The free GeoBase™ natural query language database including the commented source code on disk, ready to compile. GeoBase is a complete database designed and developed around US geography. It includes cities, mountains, rivers, and highways, and comes complete with a natural query language. Use GeoBase immediately as is, or modify it to fit your own interests.

So don't delay, don't waste a second—get Turbo Prolog today. \$99.95 is an amazingly small price to pay to become an immediate authority, an instant expert on artificial intelligence! The 21st century is only one coupon or one phone call away!

Even if you've never programmed before, our free tutorial will get you started right away.



You'll get started right away because we have included a complete step-by-step Tutorial as part of the 200-page Turbo Prolog Reference Manual. Our Tutorial will take you by the hand and teach you everything you're likely to need to know about Turbo Prolog and artificial intelligence.

For example, once you've completed the Tutorial, you'll be able to design your own "expert systems," utilizing Prolog's powerful problem-solving capabilities.

Think of Turbo Prolog as a high-speed electronic detective. First you feed it information and teach it rules. Then Turbo Prolog *thinks* the problem through and comes up with all the reasonable answers—almost instantly.

This sounds amazing, but remember Turbo Prolog is a 5th-generation language—the kind of language that 21st century computers will use routinely. In fact, you can compare Turbo Prolog to Turbo Pascal, as you would compare Turbo Pascal to machine language.



BORLAND
INTERNATIONAL
4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CALIFORNIA 95066



Bulk Rate
US Postage
PAID
Borland Int'l.

ADDRESS CORRECTION REQUESTED
FORWARDING AND RETURN POSTAGE GUARANTEED

Step-by-step Tutorial
Demo programs with Source Code included!

Borland introduces
Turbo Prolog™,
the natural language
of Artificial Intelligence.
Yours for only \$99.95

Step-by-step tutorial, demo programs with source code included!

And now Borland introduces Turbo Prolog, the natural language of Artificial Intelligence.

Prolog is probably the most powerful computer programming language ever conceived, which is why we've made it our *second* language—and "turbocharged" it to create Turbo Prolog.

Our new Turbo Prolog brings supercomputer power to your IBM® PC and introduces you step-by-step to the fascinating new world of Artificial Intelligence. And does all this for an astounding \$99.95.



Turbo Prolog is to Prolog what Turbo Pascal® is to Pascal!

Our Turbo Pascal astonished everyone who thought of Pascal as "just another language." We changed all that—and now Turbo Pascal is the de facto worldwide standard, with hundreds of thousands of enthusiasts and users in universities, research centers, schools, and with professional programmers, students, and hobbyists.

You can expect at least the same impact from Turbo Prolog, because while Turbo Prolog is the most revolutionary and natural programming language, it is also a complete development environment—just like Turbo Pascal.

Turbo Prolog radically alters and dramatically improves the brave new world of artificial intelligence—and invites you into that fascinating universe for a humanly intelligent \$99.95.

Even if you've never programmed before, our free tutorial will get you started right away

You'll get started right away because we have included a complete step-by-step tutorial as part of the 200-page Turbo Prolog Reference Manual. Our tutorial will take you by the hand and teach you everything you're likely to need to know about Turbo Prolog and artificial intelligence.

For example: once you've completed the tutorial, you'll be able to design your own expert systems utilizing Turbo Prolog's powerful problem-solving capabilities.

Think of Turbo Prolog as a high-speed electronic detective. First you feed it information and teach it rules. Then Turbo Prolog "thinks" the problem through and comes up with all the reasonable answers—almost instantly.

If you think that this is amazing, you just need to remember that Turbo Prolog is a 5th-generation language—and the kind of language that 21st century computers will use routinely. In fact, you can compare Turbo Prolog to

Turbo Pascal the way you could compare Turbo Pascal to machine language.

You get the complete Turbo Prolog programming system for only \$99.95

You get a complete Turbo Prolog development system including:

- The lightning-fast Turbo Prolog incremental compiler and the interactive Turbo Prolog editor.
- The 200-page reference manual which includes the step-by-step Turbo Prolog tutorial.
- The free GeoBase™ natural query language database including commented source code on disk—ready to compile. GeoBase is a complete database designed and developed around U.S. geography. It includes cities, mountains, rivers, and highways, and comes complete with natural query language. Use GeoBase immediately "as is," or modify it to fit your own interests.

So don't delay—don't waste a second—get Turbo Prolog now. \$99.95 is an amazingly small price to pay to become an immediate authority—an instant expert on artificial intelligence! The 21st century is only one phone call away.

YES!

Rush me Turbo Prolog Today

\$99.95

To order by phone, or for a dealer nearest you, call (800) 255-8008; in CA call (800) 742-1133

Copies	Product	Price	Totals
—	Turbo Pascal 3.0	\$69.95	—
—	Turbo Pascal w/8087	\$109.90	—
—	Turbo Pascal w/BCD	\$109.90	—
—	Turbo Pascal w/8087, BCD	\$124.95	—
—	Turbo Database Toolbox	\$54.95	—
—	Turbo Graphics Toolbox	\$54.95	—
—	Turbo Tutor	\$34.95	—
—	Turbo Editor Toolbox	\$69.95	—
—	Turbo GameWorks	\$69.95	—
—	Turbo Lightning	\$99.95	—
—	Turbo Prolog	\$99.95	—
—	Word Wizard	\$69.95	—
—	Reflex, The Analyst	\$99.95	—
—	SideKick	\$84.95	—
—	Traveling SideKick	\$69.95	—
—	SuperKey	\$69.95	—
—	Turbo Lightning Word Wizard	\$149.95	—
—	SideKick, Traveling SideKick	\$125.00	—
—	SideKick, SuperKey	\$175.00	—
—	Traveling SideKick	\$175.00	—
—	Turbo Jumbo Pack	\$245.00	—

Outside USA add \$10 per copy

CA and MA res. add sales tax

Amount enclosed

Prices include shipping to all US cities.

Carefully describe your computer system:

Mine is: ☐ 8-bit ☐ 16-bit

I use: ☐ PC-DOS ☐ MS-DOS

☐ CP/M-80 ☐ CP/M-86

My computer's name and model is:

The disk size I use is: ☐ 3 1/2" ☐ 5 1/4" ☐ 8"

Payment: ☐ VISA ☐ MC ☐ Bank Draft ☐ Check
Credit card expiration date: / /

Card #

NOT COPY-PROTECTED BI-1042
****60-DAY MONEY-BACK GUARANTEE**

Name:

Shipping Address:

City:

State: Zip:

Telephone:

CODs and purchase orders WILL NOT be accepted by Borland.

Outside USA make payment by bank draft, payable in US dollars drawn on a US bank.

*Limited Time Offer

**YES, within 60 days of purchase, if this product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund.

Turbo Prolog 1.0 Technical Specifications Programming System Features

Compiler: Incremental compiler generating native in-line code and linkable object modules. The linking format is compatible with the PC-DOS linker. Large memory model support. Compiles over 2500 lines per minute on a standard IBM PC.

Interactive Editor: The system includes a powerful interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code. At run-time, Turbo Prolog programs can call the editor, and view the running program's source code.

Type System: A flexible object-oriented type system is supported.

Windowing Support: The system supports both graphic and text windows.

Input/Output: Full I/O facilities, including formatted I/O, streams, and random access files.

Numbers Ranges: Integers: -32768 to 32767; Reals: 1E-307 to 1E+308.

Debugging: Complete built-in trace debugging capabilities allowing single-stepping of programs.



4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373

Other Borland Products include Turbo Pascal, Turbo Tutor, Turbo Lightning, Turbo Database Toolbox, Turbo Graphics Toolbox, Turbo Editor Toolbox, Turbo GameWorks, SuperKey, SideKick, The Macintosh Office Manager, Reflex, The Analyst, and Traveling SideKick—all of which are registered trademarks or trademarks of Borland International, Inc. or Borland/Analytics, Inc. Turbo Prolog and GeoBase are trademarks and Turbo Pascal is a registered trademark of Borland International, Inc. IBM and AT are registered trademarks of International Business Machines Corp. Copyright 1988 Borland International

Circle no. 161 on reader service card.



INFERENCE ENGINE

LISTING TWO (Listing continued, text begins on page 24.)

Answer is robert-sewall-ii
 No (more) answers
 which (X krystl-raquelle-brown cousin-of X)
 Answer is jonathan-wilson
 Answer is thomas-andrew-breedon
 Answer is suzanne-breedon
 Answer is amy-crider
 Answer is marty-skirvin
 Answer is rodneym-skirvin
 Answer is daren-breedon
 Answer is deidra-breedon
 Answer is deva-breedon
 Answer is stephanie-breedon
 No (more) answers

End Listing Two

LISTING THREE

((cousin-of X Y))
 ((aunt-or-uncle-of Z X))
 ((child-of Y Z))
 ((aunt-or-uncle-of X Y))
 ((parent-of Z Y))
 ((sibling-of X Z))
 ((half-sibling-of X Y))
 ((mother-of Z X))
 ((mother-of Z Y))
 ((NOT ? ((sibling-of X Y)))
 ((NOT ? ((EQ X Y))))
 ((half-sibling-of X Y))
 ((father-of Z X))
 ((father-of Z Y))
 ((NOT ? ((sibling-of X Y)))
 ((NOT ? ((EQ X Y))))
 ((male X))
 ((father-of X Y))
 ((male john-viscar))
 ((male shaun-breedon))
 ((female X))
 ((mother-of X Y))
 ((female willena-breedon))
 ((female bonnie-breedon))
 ((female karen-breedon))
 ((female betty-breedon))
 ((female judy-tracey))
 ((wife-of X Y))
 ((wife X Y))
 ((wife-of judy-tracy tommy-breedon))
 ((wife X Y))
 ((mother-of X Z))
 ((father-of Y Z))
 ((husband X Y))
 ((wife Y X))
 ((husband-of X Y))
 ((husband X Y))
 ((husband-of john-viscar pauline-davis))
 ((descendant-of X Y))
 ((child-of X Y))
 ((descendant-of X Y))
 ((child-of Z Y))
 ((descendant-of X Z))
 ((child-of X Y))
 ((parent-of Y X))
 ((parent-of X Y))
 ((father-of X Y))
 ((parent-of X Y))
 ((mother-of X Y))
 ((ancestor-of X Y))
 ((parent-of X Y))
 ((ancestor-of X Y))
 ((parent-of Z Y))
 ((ancestor-of X Z))
 ((mother-of lavenia-mccollister robert-brown-jr))
 ((father-of elijah-brown robert-brown-sr))
 ((father-of robert-brown-sr robert-brown-jr))
 ((father-of john-mccollister lavenia-mccollister))
 ((father-of isam-mccollister john-mccollister))
 ((father-of elias-presley margret-presley))
 ((father-of robert-brown-jr robert-brown-iii))
 ((father-of paul-h-sewall virginia-sewall))
 ((father-of paul-h-sewall paul-sewall-jr))
 ((father-of paul-sewall-jr peter-sewall))
 ((father-of paul-sewall-jr dee-dee-sewall))
 ((father-of paul-sewall-jr mark-sewall))
 ((father-of paul-sewall-jr paul-sewall-iii))
 ((father-of robert-brown-jr kenneth-brown))
 ((father-of robert-brown-sr amos-trice-brown))
 ((father-of robert-brown-sr james-elro-brown))
 ((father-of clarence-bailey-compton elanor-compton))
 ((father-of george-washington-compton clarence-bailey-compton))
 ((father-of samuel-compton george-washington-compton))
 ((father-of henry-sewall paul-h-sewall))
 ((father-of dr-james-d-nelson rachel-nelson))
 ((father-of robert-sewall henry-sewall))
 ((father-of paul-hebert evelina-hebert))
 ((father-of harry-breedon X))
 ((mother-of dorothy-wallace X))
 ((father-of berke-breedon harry-breedon))
 ((father-of john-wallace dorothy-wallace))

((father-of robert-brown-iii krystl-raquelle-brown))
 ((father-of paul-sewall-jr robert-sewall-ii))
 ((father-of danny-crider amy-crider))
 ((father-of bill-skirvin marty-skirvin))
 ((father-of bill-skirvin rodneym-skirvin))
 ((father-of tommy-breedon thomas-andrew-breedon))
 ((father-of tommy-breedon suzanne-breedon))
 ((father-of john-alsop joy-alsop))
 ((father-of lester-stevens geneva-stevens))
 ((father-of strawder-breedon daren-breedon))
 ((father-of daren-breedon shaun-breedon))
 ((father-of strawder-breedon deidra-breedon))
 ((father-of robert-bishop krystal-bishop))
 ((father-of robert-bishop tiffany-bishop))
 ((father-of robert-brown-sr opal-brown))
 ((father-of strawder-breedon deva-breedon))
 ((father-of strawder-breedon stephanie-breedon))
 ((father-of johnny-wilson jonathan-wilson))
 ((father-of lester-stevens geneva-stevens))
 ((mother-of bettie-holt lavenia-mccollister))
 ((mother-of miss-hornsby robert-brown-sr))
 ((mother-of elizabeth-arthur john-mccollister))
 ((mother-of margret-e-presley john-mccollister))
 ((mother-of virginia-sewall robert-brown-iii))
 ((mother-of elanor-compton virginia-sewall))
 ((mother-of lillian-givens X))
 ((father-of paul-sewall-jr X))
 ((mother-of virginia-sewall kenneth-brown))
 ((mother-of lavenia-mccollister amos-trice-brown))
 ((mother-of lavenia-mccollister james-elro-brown))
 ((mother-of sarah-virginia-sanford elanor-compton))
 ((mother-of mary-eliza-sanford clarence-bailey-compton))
 ((mother-of rachel-nelson paul-h-sewall))
 ((mother-of jane-kirk rachel-nelson))
 ((mother-of evelina-hebert henry-sewall))
 ((mother-of eugenia-hamilton evelina-hebert))
 ((mother-of dorothy-wallace darlene-breedon))
 ((mother-of dorothy-wallace willena-breedon))
 ((mother-of dorothy-wallace karen-breedon))
 ((mother-of dorothy-wallace bonnie-breedon))
 ((mother-of dorothy-wallace tommy-breedon))
 ((mother-of dorothy-wallace dorothy-breedon))
 ((mother-of dorothy-wallace brenda-breedon))
 ((mother-of dorothy-wallace betty-breedon))
 ((mother-of dorothy-wallace strawder-breedon))
 ((mother-of flo-marsh harry-breedon))
 ((mother-of christine-xxx dorothy-wallace))
 ((mother-of darlene-breedon krystl-raquelle-brown))
 ((mother-of dorothy-breedon amy-crider))
 ((mother-of brenda-breedon X))
 ((father-of bill-skirvin X))
 ((mother-of joy-alsop suzanne-breedon))
 ((mother-of joy-alsop thomas-andrew-breedon))
 ((mother-of pauline-davis joy-alsop))
 ((mother-of myrtle-jackson geneva-stevens))
 ((mother-of tammy-xxx shaun-breedon))
 ((mother-of deidra-breedon krystal-bishop))
 ((mother-of deidra-breedon tiffany-bishop))
 ((mother-of lavenia-mccollister opal-brown))
 ((mother-of lavenia-mccollister robert-brown-jr))
 ((mother-of elanor-compton paul-sewall-jr))
 ((mother-of karen-breedon jonathan-wilson))
 ((mother-of geneva-stevens X))
 ((father-of strawder-breedon X))
 ((mother-of myrtle-jackson geneva-stevens))
 ((sibling-of X Y))
 ((mother-of Z X))
 ((mother-of Z Y))
 ((father-of X X))
 ((father-of X Y))
 ((NOT ? ((EQ X Y))))

End Listing Three

LISTING FOUR

(DEFUN * (NLAMBDA X
 NIL))
 (The function (CAAR X means (CAR (CAR X)). This convention is
 extended to (CDAR X) means (CDR (CAR X)), etc. *)
 (DEFUN CONCAT (LAMBDA (P Q)
 (* concatenate 2 lists *)
 (NULL P) Q)
 (CONS (CAR P) (CONCAT (CDR P) Q)))
 (DEFUN PURGE (LAMBDA (L)
 (* purge a list of duplicate members *)
 (NULL L) NIL)
 (MEMBER (CAR L) (CDR L))
 (PURGE (CDR L)))
 (CONS (CAR L) (PURGE (CDR L))))
 (DEFUN JOIN (LAMBDA (L M)
 (* form the set union of 2 lists *)
 (PURGE (CONCAT L M)))
 (DEFUN MEET (LAMBDA (L M)
 (* form the set intersection of 2 lists *)
 (PURGE (COMMON L M))))


```

(DEFUN COMMON (LAMBDA (L M)
  (* form the set intersection of 2 lists with possible
  duplicate entries *)
  ((NULL L) NIL)
  ((MEMBER (CAR L) M)
   (CONS (CAR L) (COMMON (CDR L) M)))
  (COMMON (CDR L) M)))

(DEFUN DEF (LAMBDA (V E)
  (* determine whether the variable V is defined in the
  environment E *)
  ((NULL E) NIL)
  ((EQUAL V (CAAR E)) T)
  (DEF V (CDR E)))

(DEFUN IMM (LAMBDA (V E)
  (* return the immediate successor of the variable V in the
  environment E *)
  ((NULL E) NIL)
  ((EQUAL V (CAAR E))
   (CDAR E))
  (IMM V (CDR E)))

(DEFUN ULT (LAMBDA (V E)
  (* return the ultimate successor of the variable V in the
  environment E *)
  ((DEF V E)
   (ULT (IMM V E) E))
  V))

(DEFUN PAIRP (LAMBDA (P)
  (* determine if P is a pair (not an atom) *)
  ((ATOM P) NIL)
  T))

(DEFUN VARIABLEP (LAMBDA (V)
  (* determine if V is a variable *)
  (EQ (CAR (UNPACK V)) '*)))

(DEFUN RECREAL (LAMBDA (F E)
  (* return the recursive realization of a formula F in the
  environment E which amounts to instantiating all variables
  from the environment *)
  ((PAIRP F)
   (CONS (RECREAL (CAR F) E) (RECREAL (CDR F) E)))
  (DEF F E)

```

```

  (RECREAL (ULT F E) E)
  F))

(DEFUN OCCURS (LAMBDA (V X E)
  (* see if the variable V occurs in the term X under the
  environment E *)
  ((VARIABLEP X)
   ((DEF X E)
    (OCCURS V (IMM X E) E))
   ((EQ V X) T))
  ((ATOM X) NIL)
  (OCCURS V (CAR X) E) T)
  (OCCURS V (CDR X) E)))

(DEFUN UNIFY (LAMBDA (A B E)
  (* unify 2 expressions (A and B) in the environment E by
  extending E to the most general unifier of A and B and
  returning that environment *)
  ((EQ E 'IMPOSSIBLE)
   'IMPOSSIBLE)
  (EQUATE (ULT A E) (ULT B E) E)))

(DEFUN EQUATE (LAMBDA (A B E)
  (* auxiliary routine to UNIFY so that the flow of control
  ping-pongs recursively between UNIFY and EQUATE to
  construct the most general unifier of A and B *)
  ((EQUAL A B) E)
  ((VARIABLEP A)
   ((OCCURS A B E)
    'IMPOSSIBLE)
   (CONS (CONS A B) E))
  ((VARIABLEP B)
   ((OCCURS B A E)
    'IMPOSSIBLE)
   (CONS (CONS B A) E))
  ((ATOM A)
   'IMPOSSIBLE)
  ((ATOM B)
   'IMPOSSIBLE)
  (UNIFY (CDR A) (CDR B) (UNIFY (CAR A) (CAR B) E)))

(DEFUN VARS (LAMBDA (X)
  (* return a list of all the variables found in the expression
  X *)
  ((NULL X) NIL)
  ((VARIABLEP X) X)

```

(continued on next page)

NEON™ TURN ON THE FULL POWER OF YOUR MACINTOSH

THIS MESSAGE IS FOR BEGINNING PROGRAMMERS, TOO!

Hidden within your Mac is the programming power, flexibility, and speed to match your imagination. Neon is your key to this object-oriented world. Based on the same design philosophy as the Mac itself, Neon lets you create and command objects — program modules that you build, perfect, and add to your personal collection of tools. In this Smalltalk-like environment, you wield the power to quickly assemble and test ideas, tuning as you go for maximum speed and efficiency. □ Neon now includes full floating point support. An integrated Neon Assembler is also available as an add-on product. □ Neon is easy to work with and comes with a comprehensive manual by Danny Goodman. □ Created by Kriya Systems, Inc. for the development of our Typing Tutor III™ program, Neon is your answer for professional software development. With Neon, there are no licensing fees. Ever. For the Macintosh developer, Neon is the best choice.



To choose, call 1-800-34-KRIYA (In Virginia, 703-430-8800) with Visa/MC. Or Write Kriya Systems, Inc., Six Export Drive, Sterling, VA 22170. Neon, \$195; Neon Assembler, \$50. Add \$5 shipping for each.

Neon and Typing Tutor III are trademarks of Kriya Systems, Inc. Apple Macintosh is a trademark of Apple Computers, Inc.

Circle no. 100 on reader service card.

INFERENCE ENGINE

LISTING FOUR (Listing continued, text begins on page 24.)

```
((ATOM X) NIL)
(CONCAT (VARS (CAR X)) (VARS (CDR X))) ))

(DEFUN VARIANT (LAMBDA (Q E D)
  (* returns a variant of D such that all variables are distinct
  from those of Q in the environment E *)
  (RECREAL D (MAKENV (VARS Q) (VARS E) (VARS D))) ))

(DEFUN MAKENV (LAMBDA (Q E D)
  (* make an environment such that the instantiation of D in
  this environment will have no variables in common with the
  instantiation of Q in E *)
  (MAKENV1 (MEET D (JOIN Q E)) (JOIN (JOIN Q E) D)) ))

(DEFUN MAKENV1 (LAMBDA (P Q V)
  (* does the dirty work for MAKENV and VARIANT by generating an
  environment that sticks asterisks onto the front of all
  variable names in Q that also occur in P (V is a local
  temporary variable) *)
  ((NULL P) NIL)
  ((ATOM P)
   (SETQ V P)
   (LOOP
    ((NOT (MEMBER V Q)))
    (SETQ V (PACK (CONS '* (UNPACK V)))) )
   (CONS P V) )
  (CONS (MAKENV1 (CAR P) Q) (MAKENV1 (CDR P) Q)) ))

(DEFUN GETLIT (LAMBDA (CL N)
  (* get the Nth litteral in the clause CL and return it both as
  the functions value and in the free variable LITG along
  with its sign in the free variable SIGNG *)
  (SETQ LITG (CAR (NTH CL N)))
  (SETQ SIGNG (NOT (EQ (CAR LITG) '~)))
  ((NOT (NULL SIGNG)) LITG)
  (CADR LITG) ))

(DEFUN FRONT (LAMBDA (C N)
  (* return all elements in the clause C in front of element N
  *)
  ((NULL C) NIL)
  ((ZEROP N) NIL)
  ((EQ N 1) NIL)
  (CONS (CAR C) (FRONT (CDR C) (SUB1 N))) ))

(DEFUN BACK (LAMBDA (C N)
  (* return all elements in clause C in back of element N *)
  (NTH C (ADD1 N)) ))

(DEFUN FACTOR (LAMBDA (C P1 N1 S1 P2 N2 S2 ENV)
  (* returns a factored version of the clause C *)
  (SETQ ENV 'IMPOSSIBLE)
  (SETQ N1 0)
  (LOOP
   (SETQ N1 (ADD1 N1))
   ((GREATERP N1 (LENGTH C))
    'IMPOSSIBLE)
   (GETLIT C N1)
   (SETQ P1 LITG)
   (SETQ S1 SIGNG)
   (SETQ N2 N1)
   ( ( (LOOP
      (SETQ N2 (ADD1 N2))
      ((GREATERP N2 (LENGTH C)))
      (GETLIT C N2)
      (SETQ P2 LITG)
      (SETQ S2 SIGNG)
      ( ( (EQ S1 S2)
        (SETQ ENV (UNIFY P1 P2)) ) )
        ((NOT (EQ ENV 'IMPOSSIBLE))
         (SETQ C (RECREAL C ENV))
         (SETQ C (APPEND (FRONT C N1) (BACK C N1))) ) ) )
        ((NOT (EQ ENV 'IMPOSSIBLE)) C)
        'IMPOSSIBLE ) )
   (DEFUN BINRES (LAMBDA (CL1 N1 CL2 N2 ENV LIT1 SGN1 LIT2 SGN2 L1
    L2 LITG SIGNG)
    (* compute the binary resolvent of clause CL1 litteral number
    N1 with clause CL2 litteral number N2 and return the
    instantiated resolvent *)
    (SETQ CL2 (VARIANT CL1 ENV CL2))
    (SETQ LIT1 (GETLIT CL1 N1))
    (SETQ L1 LITG)
    (SETQ SGN1 SIGNG)
    (SETQ LIT2 (GETLIT CL2 N2))
    (SETQ L2 LITG)
    (SETQ SGN2 SIGNG)
    ((EQ SGN1 SGN2) NIL)
    (SETQ ENV (UNIFY LIT1 LIT2 ENV))
    ((EQ ENV 'IMPOSSIBLE)
     'IMPOSSIBLE)
    (RECREAL (APPEND (FRONT CL1 N1) (BACK CL1 N1) (FRONT CL2 N2)
     (BACK CL2 N2)) ENV) ))
  (DEFUN PARAMOD (LAMBDA (FROM INTO PF F PI I SEL SUB ENV SWAP PM)
    (* return the paramodulant of the FROM clause into the INTO
    clause *)
    (SETQ INTO (VARIANT FROM NIL INTO))
    (SETQ PM 'IMPOSSIBLE)
    (SETQ PF 0)
    (LOOP
     (SETQ PF (ADD1 PF))
     (SETQ F (CAR (NTH FROM PF)))
     ((NULL F))
     ((EQ (CAR F) '~)
      (SETQ SEL (CADR F))
      (SETQ SUB (CADDR F))
      (SETQ PI 0)
      (LOOP
       (SETQ PI (ADD1 PI))
       (SETQ I (CAR (NTH INTO PI)))
       ((NULL I))
       (SETQ ENV (UNIFY SEL I))
       ( ( (EQ ENV 'IMPOSSIBLE)
        (SETQ ENV (UNIFY SUB I))
        (SETQ SUB SEL) ) )
        ((NOT (EQ ENV 'IMPOSSIBLE))
         (SETQ FROM (RECREAL FROM ENV))
         (SETQ INTO (RECREAL INTO ENV))
         ( ( (NOT (NULL SWAP))
          (SETQ SUB SEL) ) )
          (SETQ INTO (APPEND (FRONT INTO PI) (LIST SUB) (BACK
           INTO PI)))
          (SETQ FROM (APPEND (FRONT FROM PF) (BACK FROM PF)))
          (SETQ PM (APPEND FROM INTO)) ) )
        ((NOT (EQ PM 'IMPOSSIBLE))) )
      PM ) )
    PM ))
```

End Listing Four

LISTING FIVE

```
(SETQ C1 '((F *X)))
((F *X))
(SETQ C2 '((G *X)))
((G *X))
(SETQ E (UNIFY C1 C2))
IMPOSSIBLE
(RECREAL C1 E)
((F *X))
(RECREAL C2 E)
((G *X))

(SETQ C2 '((F A)))
((F A))
(SETQ E (UNIFY C1 C2))
((F A))
(RECREAL C1 E)
((F A))
(RECREAL C2 E)
((F A))

(SETQ C2 '((F (G *Y))))
((F (G *Y)))
(SETQ E (UNIFY C1 C2))
((F (G *Y)))
(RECREAL C1 E)
((F (G *Y)))
(RECREAL C2 E)
((F (G *Y)))

(SETQ C2 '((F (F *X))))
((F (F *X)))
(SETQ E (UNIFY C1 C2))
IMPOSSIBLE
(RECREAL C1 E)
((F *X))
(RECREAL C2 E)
((F (F *X)))

(SETQ C1 '((F *X A) (G B *Y)))
((F *X A) (G B *Y))
(SETQ C2 '((F B *Y) (G *X *Z)))
((F B *Y) (G *X *Z))
(SETQ E (UNIFY C1 C2))
((F B *Y) (G *X *Z) (F *X A) (G B *Y))
(RECREAL C1 E)
((F B A) (G B A))
(RECREAL C2 E)
((F B A) (G B A))
```

End Listing Five

LISTING SIX

```
(SETQ CL1 '((F *X) (~ (G *Y A)) (F B)))
((F *X) (~ (G *Y A)) (F B))

(FACTOR CL1)
((~ (G *Y A)) (F B))

(SETQ CL2 '((G B *X) (F *Y)))
((G B *X) (F *Y))

(BINRES CL1 2 CL2 1)
((F *X) (F B) (F *Y))

(SETQ FROM '((F *X) (~ (G *Y A) (F *Y)) (F B)))
((F *X) (~ (G *Y A) (F *Y)) (F B))

(SETQ INTO CL2)
((G B *X) (F *Y))

(PARAMOD FROM INTO)
((F *X) (F B) (F *Y) (F *Y))
```

End Listing Six

LISTING SEVEN

```
(DEFUN VARIABLEP (LAMBDA (V)
  (* determine if V is a variable: if it starts with lower-case
  it is *)
  ((ATOM V)
   (GREATERP (ASCII (CAR (UNPACK V))) (ASCII ')))
  NIL))

(DEFUN LPAR (LAMBDA (R)
  (* return left parent *)
  (CAR R))

(DEFUN LLIT# (LAMBDA (R)
  (* return left litteral number *)
  (CADR R))

(DEFUN RPAR (LAMBDA (R)
  (* return right parent *)
  (CADDR R))

(DEFUN RLIT# (LAMBDA (R)
  (* return right litteral number *)
  (CAR (CDDDR R)))

(DEFUN NLITS (LAMBDA (R)
  (* return number of litterals *)
  (CADR (CDDDR R)))

(DEFUN MAXNDX (LAMBDA (R)
  (* return maximum index *)
  (CADDR (CDDDR R)))

(DEFUN BINDINGS (LAMBDA (R)
  (* return the bindings *)
  (CAR (CDDDR (CDDDR R))))

(DEFUN SETLPAR (LAMBDA (R V)
  (* set left parent *)
  (RPLACA R V))

(DEFUN SETLLIT# (LAMBDA (R V)
  (* set left litteral number *)
  (RPLACA (CDR R) V))

(DEFUN SETRPAR (LAMBDA (R V)
  (* set right parent *)
  (RPLACA (CDDDR R) V))

(DEFUN SETRLIT# (LAMBDA (R V)
  (* set right litteral number *)
  (RPLACA (CDDDR R) V))

(DEFUN SETNUMLITS (LAMBDA (R V)
  (* set number of litterals *)
  (RPLACA (CDR (CDDDR R)) V))

(DEFUN SETMAXNDX (LAMBDA (R V)
  (* set maximum index *)
  (RPLACA (CDDDR (CDDDR R)) V))

(DEFUN SETBINDINGS (LAMBDA (R V)
  (* set bindings *)
  (RPLACA (CDDDR (CDDDR R)) V))

(DEFUN INRECP (LAMBDA (R)
  (* is R an input record? *)
  (NULL (RPAR R)))

(DEFUN LEQP (LAMBDA (X Y)
  (* is X less than or equal to Y ? *)
  (NOT (GREATERP X Y)))

(DEFUN NMEMS (LAMBDA (L)
  (* return the number of members in the list L *)
  ((NULL L) 0)
  (ADD1 (NMEMS (CDR L))))

(DEFUN EXTRACT (LAMBDA (K L TMP)
  (* return the Kth member of L *)
  (* TMP is a local variable *)
  (LOOP
   ((ZEROP K) TMP)
   (SETQ K (SUB1 K))
   (SETQ TMP (POP L))))

(DEFUN RESOLVE (LAMBDA (CL1 I CL2 J LLIT RLIT LNDX RNDX BNDEV
  LSIGN RSIGN)
  (* resolve clause CL1 litteral I with clause CL2 litteral J
  returning a new clause record representing the resolvent:
  UNIFY will extend the binding environment: returns NIL if
  impossible *)
  (GETLIT CL1 I)
  (SETQ LLIT LITG)
  (SETQ LNDX INDEXG)
  (SETQ LSIGN SIGNG)
  (GETLIT CL2 J)
  (SETQ RLIT LITG)
  (SETQ RNDX (PLUS INDEXG (MAXNDX CL1)))
  (SETQ RSIGN SIGNG)
  (* create the new clause record *)
  (SETQ BNDEV (LIST CL1 I CL2 J (DIFFERENCE (PLUS (NLITS CL1)
  (NLITS CL2)) 2) (PLUS (MAXNDX CL1) (MAXNDX CL2)) NIL))
  (* test for opposite signs *)
  ((EQ LSIGN RSIGN) NIL)
  (* extend the environment by the unification algorithm *)
  ((UNIFY LLIT LNDX RLIT RNDX) BNDEV)
  NIL))

(DEFUN GETLIT (LAMBDA (CL K)
  (* return the Kth member of CL *)
  (GETLIT CL K)))
```

(continued on next page)

Wizard C

Discover the powers of Wizard C
for yourself!

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language
February, 1985

"Wizard's got the highest marks for support."

"The Wizard compiler had excellent diagnostics; it would be easier writing portable code with it than with any other compiler we tested."

Dr. Dobb's Journal
August, 1985

Full Lint checking, six memory models, 8087 support, in-line assembly language, ROMable data support, full library source code. Cross-compilers are available on VAX/VMS and UNIX machines.

(617) 641-2379

Only \$450.



SYSTEMS SOFTWARE, INC.

Circle no. 116 on reader service card.

11 Willow Court
Arlington, MA 02174



RECOVER ERASED FILES
REPAIR DAMAGED FILES

\$69.95

File Recovery System

Recover a file that was erased
Repair files that 'won't load'
Edit a file or anything on the disk
Context-Sensitive Help
Change File Attributes
'Wipe' Feature

Call Now, 7 Days A Week, 24 Hours A Day
(408) 395-9568



"... More powerful than the
Norton Utility Version 3.1."
—PC Magazine

15100 EL CAMINO GRANDE,
SARATOGA, CA 95070

©1986 Software Resource Group, Inc. All rights reserved.

Circle no. 267 on reader service card.

INFERENCE ENGINE

LISTING SEVEN (Listing continued, text begins on page 24.)

```
(* get the Kth literal in clause CL: return the literal
gotten in LITG: and the associated index in INDEXG *)
(* if CL is an input clause then extract the Kth literal and
set the index to 1 *)
((INRECP CL)
 (SETQ LITG (EXTRACT K (LPAR CL)))
 (SETQ SIGNG (EQ K 1))
 (SETQ INDEXG 1) )
(* if it is in the left parent of the clause then get the
literal from the left parent *)
(* this is true if K is less than the literal last resolved
upon in the left parent of the current clause *)
((LESSP K (LLIT# CL))
 (GETLIT (LPAR CL) K) )
(* this is also true if K is less than the number of literals
in the left parent but in this case we must adjust K by 1
*)
((LESSP K (NUMLITS (LPAR CL)))
 (GETLIT (LPAR CL) (ADD1 K)) )
(* if the selected literal is in the right parent but left of
the literal last resolved upon then get the literal from
the right parent with the appropriate adjustment to K *)
((LESSP K (PLUS (SUB1 (NUMLITS (LPAR CL))) (RLIT# CL)))
 (GETLIT (RPAR CL) (ADD1 (DIFFERENCE K (NUMLITS (LPAR CL)))))
 (* in this case adjust the index got *)
 (SETQ INDEXG (PLUS INDEXG (MAXNDX (LPAR CL)))) )
(* otherwise the selected literal is in the right parent to
the right of last literal resolved upon so adjust K
accordingly and get the literal *)
(GETLIT (RPAR CL) (PLUS (DIFFERENCE K (NUMLITS (LPAR CL))) 2))
 (* and adjust the index gotten *)
 (SETQ INDEXG (PLUS INDEXG (MAXNDX (LPAR CL))))) )

(DEFUN UNIFY (LAMBDA (TERM1 INDEX1 TERM2 INDEX2)
 (* attempt to unify TERM1 under INDEX1 with TERM2 under INDEX2
and extend the binding environment represented in the
global variable BNDEV: return T if successful or NIL if the
unification is impossible *)
 (* if both terms and indices are equal then return T: no
extension to BNDEV is needed *)
 (EQUAL TERM1 TERM2)
 (EQ INDEX1 INDEX2)
 T )
 (* if TERM1 is a variable *)
 ((VARIABLEP TERM1)
 (* then if it is bound in the current environment *)
 ((ISBOUND TERM1 INDEX1 BNDEV)
 (* then substitute that binding and attempt to unify again
*)
 (UNIFY TERM2 INDEX2 TERM2 INDEX2) )
 (* else if the variable of TERM1 occurs in TERM2 then we
have a recursive "black" "hole" situation so return
NIL *)
 ((OCCUR TERM1 INDEX1 TERM2 INDEX2) NIL)
 (* else force a unification by adding the necessary binding
and return T for success *)
 (BIND TERM1 INDEX1 TERM2 INDEX2 BNDEV)
 T )
 (* if TERM2 is a variable then swap the 2 terms and UNIFY the
other way *)
 ((VARIABLEP TERM2)
 (UNIFY TERM2 INDEX2 TERM1 INDEX1) )
 (* otherwise if the heads of the terms unify then return the
result of unifying the tails of the terms: the environment
is extended as needed *)
 ((UNIFY (CAR TERM1) INDEX1 (CAR TERM2) INDEX2)
 (UNIFY (CDR TERM1) INDEX1 (CDR TERM2) INDEX2) ) ) )

(DEFUN ISBOUND (LAMBDA (VAR INDEX BNDEV)
 (* determine if the variable VAR under the index INDEX is
bound in the binding environment BNDEV: if it is then
return T and set the free variables TERMB and INDEXB to
the term and index respectively to which it is bound: *)
 (* otherwise return NIL and do not alter the values of TERMB and
INDEXB *)
 (* if BNDEV is an input record then it cannot be bound so
return NIL *)
 ((INRECP BNDEV) NIL)
 (* if VAR under INDEX is equal to the head of the binding
environment at this level then return T and set TERMB and
INDEXB accordingly *)
 ((EQUAL (CONS VAR INDEX) (CAR (BINDINGS BNDEV)))
 (SETQ TERMB (CADAR (BINDINGS BNDEV)))
 (SETQ INDEXB (CAR (CDDAR (BINDINGS BNDEV))))
 T )
 (* else see if it is bound in the tail of the binding
environment at this level *)
 ((ISBOUND VAR INDEX (CDR (BINDINGS BNDEV))) T)
 (* if not then check INDEX to see whether to search the left
or right parent binding environment *)
 ((LEQP INDEX (MAXNDX (LPAR BNDEV)))
 (* search left parent *)
 (ISBOUND VAR INDEX (LPAR BNDEV)) )
 (* search right parent *)
 ((ISBOUND VAR (DIFFERENCE INDEX (MAXNDX (LPAR BNDEV))) (RPAR
BNDEV))
 (* adjust INDEXB accordingly *)
 (SETQ INDEXB (PLUS INDEXB (MAXNDX (LPAR BNDEV))))
 (* and return success *)
 T )
 (* all possible approaches failed so return NIL for not bound *)
 NIL ) )

(DEFUN OCCUR (LAMBDA (V I TERM J)
 (* see if the variable V under the index I occurs in the term
```

```
TERM under the index J and return T or NIL *)
(* if TERM is a variable *)
((VARIABLEP TERM)
 (* then if it is bound *)
 ((ISBOUND TERM J BNDEV)
 (* then make the substitution and test for occurrence *)
 (OCCUR V I TERMB INDEXB) )
 (* if V equals TERM *)
 ((EQ V TERM)
 (* then return T if I=J else NIL *)
 (EQ I J) ) )
(* if TERM is atomic and not a variable *)
(* then it is a constant so return NIL *)
((ATOM TERM) NIL)
(* otherwise if V under I occurs in the head of TERM under J
then return T *)
((OCCUR V I (CAR TERM) J) T)
(* otherwise return T if V under I occurs in the tail of TERM
under J and NIL otherwise *)
(OCCUR V I (CDR TERM) J) ) )

(DEFUN BIND (LAMBDA (V I TERM J BNDEV)
 (* bind V under I to TERM under J in BNDEV *)
 (SETBINDINGS BNDEV (CONS (CONS (CONS V I) (CONS TERM J))
 (BINDINGS BNDEV)))) )

(DEFUN * (LAMBDA COMMENTS
 NIL ) )

(DEFUN MAKECL (LAMBDA (CL)
 (* make a clause record out of the expression CL *)
 (LIST CL 0 NIL 0 (NMEMS CL) 1 NIL) ) )
```

End Listing Seven

LISTING EIGHT

```
(SETQ CLAUSE-1 '((F x y) (G x) (P A y x) ))
((F x y) (G x) (P A y x))

(SETQ CLAUSE-2 '((P A B C)))
((P A B C))

(SETQ CLAUSE-3 '((G C)))
((G C))

(SETQ TEST (RESOLVE (MAKECL CLAUSE-1) 2 (MAKECL CLAUSE-3) 1))
(
  ( ((F x y) (G x) (P A y x)) 0 NIL 0 3 1 NIL )
  2
  ( ((G C)) 0 NIL 0 1 1 NIL )
  1
  2
  2
  ( ((x . 1) C . 2) )
)
```

End Listings

PRIME FEATURES

- Execute DOS level commands in HS/FORTH, or execute DOS and BIOS functions directly.
- Execute other programs under HS/FORTH supervision.
(editors debuggers file managers etc)
- Use our editor or your own.
- Save environment any time as .COM or .EXE file.
- Eliminate headers, reclaim space without recompiling.
- Trace and decompile.
- Deferred definition, execution vectors, case, interrupt handlers.

HS/ FORTH

- Full 8087 high level support.
Full range transcendentals
(tan sin cos arctan logs exponentials)
- Data type conversion and I/O parse/format to 18 digits plus exponent.
- Complete Assembler for 8088, 80186, and 8087.
- String functions -
(LEFT RIGHT MID LOC COMP XCHG JOIN)
- Graphics & Music
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management
- Full megabyte - programs or data
- Fully Optimized & Tested for:
IBM-PC XT AT and JR
COMPAQ and TANDY 1000 & 2000
(Runs on all true MSDOS compatibles!)
- Compare

BYTE Sieve Benchmark jan 83
HS/FORTH 47 sec BASIC 2000 sec
with AUTO-OPT 9 sec Assembler 5 sec
other Forths (mostly 64k) 55-140 sec
FASTEST FORTH SYSTEM

AVAILABLE.

**TWICE AS FAST AS OTHER
FULL MEGABYTE FORTHS!**

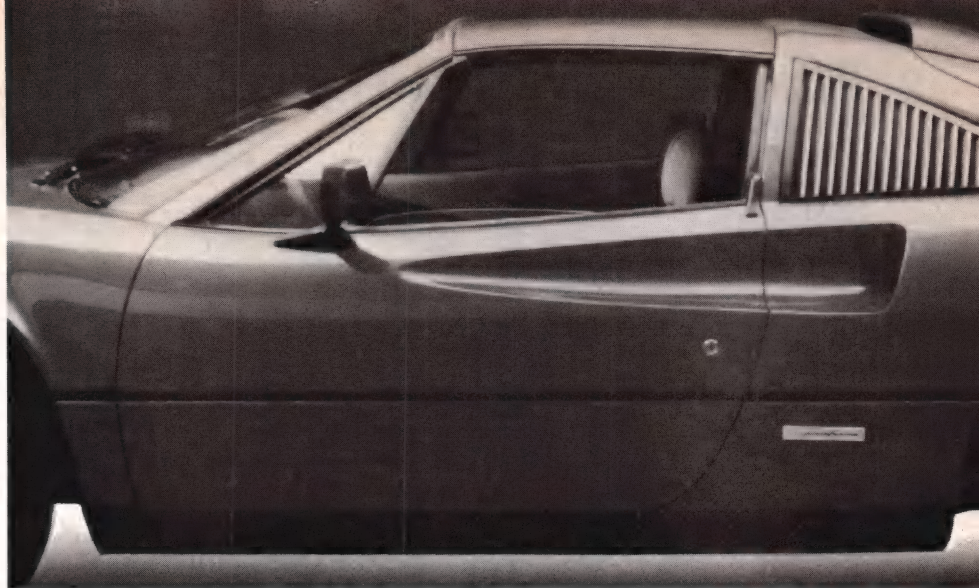
(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: \$270.



HARVARD SOFTWARES

P.O. BOX 69
SPRINGBORO, OH 45066
(513) 748-0390



Introducing The Perfect Linker For Large, Complex Programs.

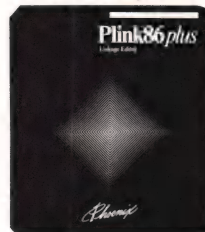
Writing MS-DOS®/PC DOS programs with dozens of modules? Then, you need the new Plink™86 Plus high-performance linkage editor.

You can create up to 4,095 overlays, in one or many files, on one or many disks. And in any MS-DOS/PC DOS environment Plink 86 Plus will automatically cache overlays using all or part of available memory including IBM AT™ extended memory. Allowing you to get better performance from your large machines.

Perform an "object merge" to combine multiple object files into one. That makes it easier to link object code with application software. Harder for pirates to figure out how your software works. And, speeds up the linking process.

Reload overlays upon function return to avoid calls from one overlay "smashing" calls from another overlay. Allocate library modules to overlay structures automatically. So that modules are parceled out to where they are needed in the overlay structure, reducing the run-time size of the program.

With Plink 86 Plus you can work on modules individually. Link them into executable files. Or, use the same module in different programs. You can change the overlay structure of an existing program without recompiling. Or, use one overlay to access code and data in other overlays.



\$495 complete.

Now for a limited time, send us your old Plink 86 and \$150.00 and we'll send you the new Plink 86 Plus.

Send for your Plink 86 Plus information kit today. Call or write: Phoenix Computer Products Corporation, 320 Norwood Park South, Norwood, MA 02062 (800) 344-7200. In Massachusetts (617) 762-5030.

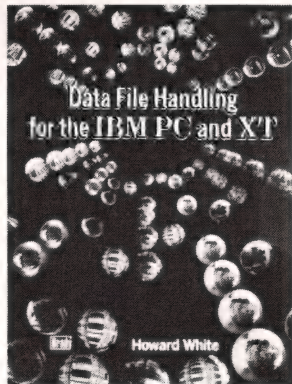
Programmers' Pfantasies
by

Phoenix

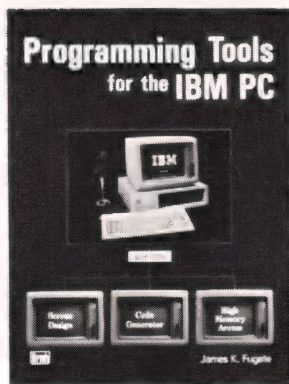
Plink 86 Plus is a trademark of Phoenix Software Associates, Ltd.
MS-DOS is a registered trademark of Microsoft Corporation.
AT is a trademark of International Business Machine Corporation.

BRADY Knows Programming.

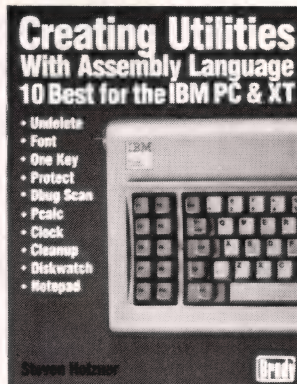
You'll learn to whip every element of your programming into shape with the latest information and guidance by America's foremost technical experts. Just call toll-free or use the coupon below to order today.



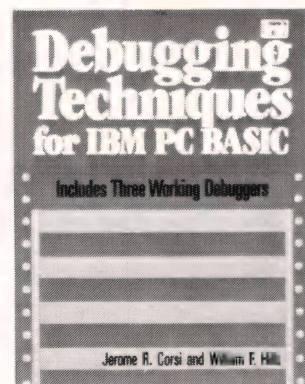
1. Take your programs to the next step... data input/output and manipulation. Learn about data storage and access and create your own database and file managers. Assumes some knowledge of BASIC. \$17.95



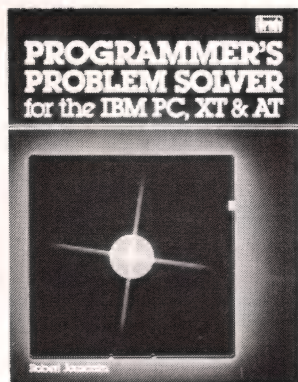
2. Includes a listing for the remarkable QUICKCODE code generator which automatically writes input, locate, and print statements. Also unleashes high memory access and makes screen design a cinch. \$19.95



3. Learn the techniques used for creating assembly language utilities. Includes 10 of the most popular utilities such as DBUG, SCAN, CLOCK, UNDELETE, ONE KEY, PCALC calculator and notepad and five others. \$21.95 (Disk available)



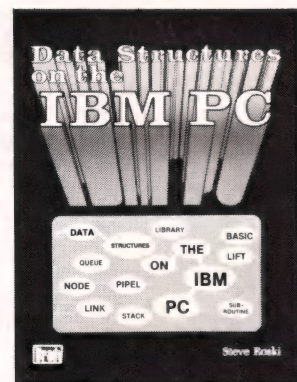
4. Includes code listings for three working debuggers including single-stepping, cross referencing, and mapping utilities. \$19.95 (Disk available)



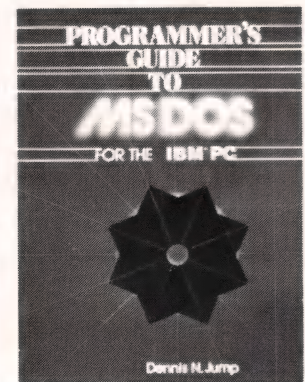
5. A definitive reference text for advanced programmers. You'll find over 150 discussions of common hardware-control tasks (in BASIC, Pascal, or C) as well as assembler overlays, drivers, and real-time operators. \$22.95



6. Probes the inner workings of the 8086 (used by the AT & T 6300) and 8088 (IBM PC) chips... and describes specific techniques for using the full capability of these chip designs while programming in assembler. \$18.95



7. Here's a compendium of many of the most useful, but often neglected, advanced programming concepts. A tutorial in format that uses BASIC for examples. It covers techniques such as: linked data structures; recursion; pipelining; and dynamic storage allocation. Includes listings for 25 sub-routines. \$21.95 (Disk available)



8. A breakthrough explanation to the technical operations of DOS for programmers. Shows how to use the I/O services with discussions of: character and non-character functions; directory and file management routines; and memory management. Includes BIOS functions and info on IBM "compatibles." \$16.95

Now at your book or computer store. Or order toll-free today: **800-624-0023** In New Jersey: 600-624-0024

BRADY COMMUNICATIONS COMPANY, INC.
c/o Prentice Hall,
P.O. Box 512, W. Nyack, NY 10994

Circle the numbers of the titles you want below.
(Payment must be enclosed; or, use your charge card.) Add \$1.50 for postage and handling.
Enclosed is check for \$_____ or charge to
☐ MasterCard ☐ VISA.

1 (0-89303-528-9)
5 (0-89303-787-7)

2 (0-89303-784-2)
6 (0-89303-424-X)

3 (0-89303-584-X)
7 (0-89303-481-9)

4 (0-89303-587-4)
8 (0-8359-5655-5)

Acc't # _____ Exp. date _____
Signature _____
Name _____
Address _____
City _____ State _____ Zip _____
(New Jersey residents, please add applicable sales tax.) GR - PTEC - AT(O)
Dept. 3



¿C? ¡SI!

If you're a C programmer (or want to be one), we speak your language. Subscribe to **The C Journal** today, and start increasing your productivity right away. We give you information you can **use** on any machine — IBM PC™, UNIX™-based, Macintosh™, or CP/M™ — micro, mini, or mainframe.

- in-depth reviews and feature articles — C compilers, editors, interpreters, function libraries, and books.
- hints and tips — help you work **better** and **faster**.
- interviews — with software entrepreneurs that **made it** — by using C.
- news and rumors — from the ANSI standards committee and the industry.

Limited Time Offer

Join our thousands of subscribers at the **Discount Rate** of only \$18 for a full year (regularly \$28)! Call us now at (201) 989-0570 for faster service — don't miss a single issue of **The C Journal**!

Please add \$9 for overseas airmail.

Trademarks — CP/M: Digital Research Inc. IBM PC: IBM Corp. Macintosh: Apple Computer Corp. **The C Journal**: InfoPro Systems. UNIX: AT&T Bell Labs.



InfoPro Systems
3108 Route 10
Denville, NJ 07834
(201) 989-0570



Circle no. 194 on reader service card.

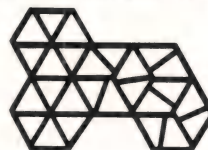
BYSO™ LISP

A production LISP compiler designed for optimum performance on the IBM PC.

- Includes interpreter — debug programs instantly.
- Compatible with Common LISP and some earlier dialects.
- Access to all system functions of IBM PC.
- Complete — has full screen editor, only PC DOS required.
- Tested and reliable — in use since Aug. '84, all known bugs fixed.
- Very fast — does Jul. '84 BYTE Sieve test in .8 seconds. Other LISP's take 30 to 80 seconds. Garbage collects in under a quarter second.
- Visual Syntax™ — an editor written in BYSO LISP that displays programs graphically.
- Fully documented — all features explained, application notes given, full source code of Visual Syntax.
- Not copy protected, preinstalled.
- Generates stand alone code that is royalty free in most cases.
- Priced as a good C compiler, not as a miracle expert system.

Interpreter only, single user license \$150.

Compiler and interpreter, single user license \$395.
Requires 256K, IBM PC or true compatible.



LEVIEEN INSTRUMENT CO.

Sittington Hill/P.O. Box 31
McDowell, VA 24458
(703) 396-3345

BYSO is a trademark of Raphael L. Levien. IBM PC is a registered trademark of the IBM Corporation.

Circle no. 252 on reader service card.

Another in a series of productivity notes on MS-DOS™ software from UniPress.

Subject: Low-cost, powerful multi-user Data Management System with keyed B+ tree Isam file manager, report writer and query facility. Runs on the IBM-PC™

PHACT was designed specifically for the programmer. PHACT provides a one-product solution for accessing and manipulating records, generating reports, and selecting data through a powerful query facility.

Features:

- Supports fixed and variable length records (1-9999 bytes).
- Up to 9 alternate indices can be utilized.
- Record locking for multiple simultaneous updates. Handles Networks!
- Full or partial key record access.
- Automatic key compression to save disk space and improve performance.
- Databases are password protected and may be opened for shared or exclusive use.
- Can be linked with these C libraries: Lattice™, Microsoft, C86, and Aztec™
- PHACT-report provides an easy to use command language for formatting reports from existing PHACT databases.
- PHACT-query uses a SEQUEL-like relational query language for selecting data.
- PHACT Forms Generator coming soon!
- Excellent product to integrate into your own packages. Call for terms on distribution rights for source purchases.

Price:

PHACT Data Management System —

	Binary	Source
Single-user system	\$ 655	\$2085
Multi-user system	1220	4985

Available for UNIX™ and VMS™ too!
Mastercard/Visa accepted

DATA MANAGEMENT SYSTEM

PHACT™

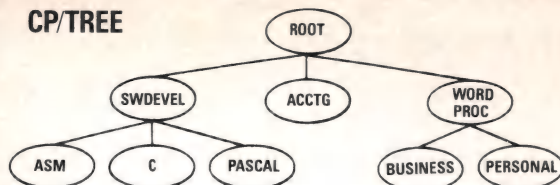
For our Free Catalog and more information on these and other MS-DOS software products, call or write:
UniPress Software Inc.
2025 Lincoln Highway
Edison, NJ 08817
Telephone: 201-985-8000
Order Desk: 800-222-0550 (Outside NJ)
Telex: 709418
European Distributor
Modulator SA
Switzerland, Telephone: 41 31 59 22 22
Telex: 911859
Japanese Distributor
D.I.T. Inc.
Minato-Ku, Tokyo
Telephone: 03 586-6580

PHACT is a trademark of PHACT Associates. IBM-PC, International Business Machines Corp., Lattice is a trademark of Lattice, Inc., MS-DOS is a trademark of Microsoft, Mark Williams is a trademark of Mark Williams, Aztec is a trademark of Aztec.

UniPress Software
Your Leading Source for UNIX™ Software.

Circle no. 77 on reader service card.

CP/TREE



Tree-Structured Named Directories for CP/M 2.2

- Transforms user areas into Unix-like directories
- Provides Unix-like directory commands:
CD, MKDIR, PWD, RMDIR, TREE
- Includes a CCP replacement featuring:
 - Command and file search path. Use programs like WordStar from any directory!
 - Erase with query
 - Wildcard rename with query
- Provides output redirection to disk file
- Uses as little as 1/2 k RAM, never more than 2 1/4 k
- A must for hard disks
- Installs easily: requires no modifications to BDOS or BIOS
- Requires standard CP/M 2.2 (not 3.0 or Apple), Z80, 48k RAM

\$29.95 plus \$4.00 s&h

To order: Specify disk format (8" SSD, NorthStar DD. Call for info on others.). MC, Visa, COD (add \$1.90), check (delays shipping 2 weeks). MA residents add 5% sales tax. POs not accepted.

Precise Electronics
P.O. Box 339
New Town Branch
Boston, MA 02258
tel: (617) 332-3977

Apple® Apple Computer. CP/M® Digital Research. WordStar® MicroPro International. Z80® Zilog. Unix® AT&T Technologies.

EXPERT-2

LISTING ONE (Text begins on page 42.)

```

\ LIFE in Expert-2

\ simple demo program - this version on MVP Forth
\ by Jack Park 1985

: WALL ; \ something to forget when done

NOSHOW \ a word added to EXPERT-2 to cause suppression of display
\ of any inferences. Sets a variable to 00. Variable is tested
\ by each printing word.

VARIABLE ARRAY1 510 ALLOT
VARIABLE ARRAY2 510 ALLOT
\ during a given pass through the cells, one array will be the
\ "old" array, the other the "new" array. On the next pass,
\ arrays reverse position.
: CLEAR1 ARRAY1 512 ERASE ;
: CLEAR2 ARRAY2 512 ERASE ;

VARIABLE ^OLD
VARIABLE ^NEW
VARIABLE ^CELL
VARIABLE ?CELL
VARIABLE CELLTOGGLE
VARIABLE II \ miscellaneous variable use in counting
VARIABLE JJ
VARIABLE KK

219 CONSTANT SYMBOL \ graphics symbol for IBM PC display
\ this symbol can be changed to virtually any ASCII symbol
\ e.g. ASCII * CONSTANT SYMBOL will print a "*" at each live cell

: IJ ( J I -- ) 32 * SWAP 2 * + ^OLD @ + @ ( is alive? )
  IF 1 ^CELL @ + ! THEN ; \ printing symbol is truth value here
\ if a printing symbol is in a cell, it is alive.
\ if a cell is alive, increment count in center cell. Note, this
\ routine counts total of alive "nearest neighbors" to center
\ cell.

: FIX ( n -- n ) DUP -1 -
  IF DROP 15
  ELSE DUP 16 - IF DROP 0 THEN
  THEN ; \ bounds checking for array edges
\ this form of bounds checking forces a square (flat) array to
\ behave like a torus - there will be end effects when a
\ life form grows beyond the visible edge of the array.

: SETCELL ( J I -- ) 32 * SWAP 2 * + ^NEW @ + 0 OVER !
  ( clear cell ) ^CELL ! ( save cell address ) ;
\ support for numeric processing of cell counts

: DOCELLS 16 0 ( -- ) \ here is the main numeric loop
  DO 16 0 ( note: 16 x 16 array of cells )
    DO J I SETCELL
      J 1- FIX I IJ
      J 1+ FIX I IJ
      J I 1- FIX IJ
      J I 1+ FIX IJ
      J 1- FIX I 1- FIX IJ
      J 1- FIX I 1+ FIX IJ
      J 1+ FIX I 1- FIX IJ
      J 1+ FIX I 1+ FIX IJ
    LOOP
  LOOP ; \ count all alive cells around each cell

\ count is saved in "NEW" cell
\ this routine could be sped up, but it runs in about 2 seconds
\ as is.

: (INITCELL) ( y x -- ) 32 * SWAP 2 * + ARRAY1 + SYMBOL SWAP ! ;

: EATER ( a starting design ) CLEAR1
  5 4 (INITCELL) 6 4 (INITCELL) 1 5 (INITCELL) 2 5 (INITCELL)
  4 5 (INITCELL) 7 5 (INITCELL) 1 6 (INITCELL) 2 6 (INITCELL)
  5 6 (INITCELL) 6 6 (INITCELL) ;

: PENTA ( a starting design ) CLEAR1
  4 6 (INITCELL) 9 6 (INITCELL) 2 7 (INITCELL) 3 7 (INITCELL)
  9 5 DO I 7 (INITCELL) LOOP 10 7 (INITCELL) 11 7 (INITCELL)
  4 8 (INITCELL) 9 8 (INITCELL) ;
\ to run the system, one types PENTA RUN, or EATER RUN
\ consult BYTE Magazine, December 1978 for further details
\ cells will not necessarily behave as advertised because of
\ edge effects in a limited array

: SHOWCELLS HOME ( alias: PAGE, clearscreen ) 16 0
  DO 16 0
    DO J 32 * I 2 * + ^NEW @ + @ EMIT LOOP CR
  LOOP CR KK @ . ; \ display the array

: RUN ( the main word ) CLEAR2 1 CELLTOGGLE !
  
```

**"Revelation is a visionary product.
It brings out the creativity
in programmers."**

*Steve Perry
Mainframe Micros, Inc.
Professional Programmers*

With Revelation, Steve Perry put a mainframe personnel management system on a micro. Now it's an integral part of the Human Resource Management System sold by Control Data Business Centers.

Revelation's capabilities go beyond ordinary DBMS's. Its sophisticated dictionary, mainframe communications and advanced report writer give more power to programmers.

Write your next application the right way. Send today for free technical specifications. Or \$24.95 gets you a complete Revelation Demo/Tutorial.



COSMOS™

Cosmos, Inc., 19530 Pacific Highway South, Suite 102
Seattle, WA 98188, (206) 824-9942, Telex: 9103808627


```

\ be sure to call one of the starting patterns before RUN
ARRAY1 ^NEW ! 0 ?CELL ! 32 0 ( run up to 32 generations )
DO 16 0 I !+ KK ! SHOWCELLS
DO I JJ ! 16 0
DO I II ! DIAGNOSE ( run the rules ) LOOP
LOOP 0 ?CELL ! ?TERMINAL IF LEAVE THEN ( tap any key to stop )
LOOP 1 KK +! SHOWCELLS ;
\ II, JJ, and KK carry loop counters outside the loops. It is
\ not possible to simply pass these values on the stack, because
\ they are used well into the DIAGNOSE - inference engine -
\ routine.

: RUNCELLS ( used by rules ) ?CELL @ NOT ( have we run yet? )
IF CELLTOGGLE @
IF ARRAY1 ^OLD ! ARRAY2 ^NEW ! 0
ELSE ARRAY2 ^OLD ! ARRAY1 ^NEW ! 1
THEN CELLTOGGLE ! DOCELLS ( get all the counts )
THEN 1 ?CELL ! ;

: (ADDR) JJ @ 32 * II @ 2* + ; \ numeric support

\ following are antecedent numeric tests used by the rules
: COUNT=0 (ADDR) ^NEW @ + @ 0= ; \ return truth to rules
: COUNT=1 (ADDR) ^NEW @ + @ 1 = ;
: COUNT=2 (ADDR) ^NEW @ + @ 2 = ;
: COUNT=3 (ADDR) ^NEW @ + @ 3 = ;
: COUNT=4 (ADDR) ^NEW @ + @ 4 < NOT ;
: ?ALIVE ( -- tf ) RUNCELLS (ADDR) ^OLD @ + @ ;
\ note use of the print character as a truth flag in ?ALIVE.
\ each antecedent test returns a truth value based on a test:
\ e.g. COUNT=0 looks at the "current" new cell to see what the
\ count of its nearest neighbors has been found to be. Returns
\ TRUE if count = 0, otherwise returns FALSE. This value is
\ the truth value for the clause that called COUNT=0 in the
\ rules (ANDRUN COUNT=0, etc.)

\ following are consequent numeric procedures called by the
\ rules
: LIVE SYMBOL (ADDR) ^NEW @ + ! TRUE ( dummy truth value ) ;
: DIE 0 (ADDR) ^NEW @ + ! TRUE ;
\ note the use of SYMBOL as a truth value; SYMBOL must be > 0
: PROPAGATE (ADDR) ^OLD @ + @ (ADDR) ^NEW @ + ! TRUE ;
\ notice that all procedures must return a truth value to
\ the inference engine - even in the consequent fields.
\ e.g. LIVE stores the SYMBOL (which means the cell is now
\ alive) into the current cell, then returns a dummy TRUE.

\ following is the knowledge base
RULES \ beginning of rules, start the rule compiler
IFRUN ?ALIVE
ANDRUN COUNT=2
THEN cell lives
ANDTHENRUN LIVE
IFRUN ?ALIVE
ANDRUN COUNT=3
THEN cell lives
ANDTHENRUN LIVE
IFNOTRUN ?ALIVE
ANDRUN COUNT=3
THEN cell lives
ANDTHENRUN LIVE
IFRUN COUNT=0
THEN cell dies
ANDTHENRUN DIE
IFRUN COUNT=4
THEN cell dies
ANDTHENRUN DIE
IFNOT cell lives
ANDNOT cell dies
THENHYP cell propagates
ANDTHENRUN PROPAGATE
DONE \ tidy up and stop the rule compiler.
\ note that EXPERT-2 inference engine must be modified with

\ addition of a variable to suppress printing out inferences.

```

End Listing



to



the dBase translator

- dBase produces quality C direct from dBase II or III programs.
- Move dBase programs to UNIX or other machines.
- Improve program speed and reliability.
- Support multi-user/network applications.
- With power guidebook of conversion hints.
- Includes full screen handler and uses your current C database manager.
- May be used to move existing programs or help dBase programmers learn C easily.
- For MSDOS, PCDOS, UNIX, XENIX, Macintosh, AMIGA. (Uses ANSI.SYS driver on MSDOS, CURSES under UNIX)
- Priced from \$350, also available from distributors.

Desktop A.I.

1720 Post Rd. E. #3
Westport, CT 06880
(203) 255-3400

Circle no. 258 on reader service card.

**"Revelation is 50% harder
to learn than dBase IIITM,
and 5 times more powerful!"**

*John Dunbar
Dunbar and Company
Application Designers*

That's a small price to pay for power. Especially when it helps a programmer like John Dunbar create LAN databases for the Houston Rockets, Compaq Computer Corporation and Houston Natural Gas.

Revelation has all the tools Dunbar needs including a program generator to slash development time and a robust language that puts C to shame.

Prove it to yourself. The Revelation Demo/Tutorial is only \$24.95. Technical specifications are free. Send today for more information.



COSMOSTM

Cosmos, Inc., 19530 Pacific Highway South, Suite 102
Seattle, WA 98188, (206) 824-9942, Telex: 9103808627
dBase III is a trademark of Ashton-Tate.

Circle no. 283 on reader service card.

68K ASSEMBLER

LISTING ONE (Text begins on page 52.)

```
DEFINITION MODULE LongNumbers;
(* Routines to handle HEX digits for the X68000 cross assembler. *)
(* All but LongPut and LongWrite are limited to 8 digit numbers. *)

FROM Files IMPORT
FILE;

EXPORT QUALIFIED
LONG, LongClear, LongAdd, LongSub, LongInc, LongDec,
LongCompare, CardToLong, LongToCard, LongToInt,
LongPut, LongWrite, StringToLong, AddrBoundL, AddrBoundW;

CONST
DIGITS = 8;
BASE = 16;

TYPE
LONG = ARRAY [1..DIGITS] OF INTEGER;

PROCEDURE LongClear (VAR A : LONG);
(* Sets LONG to Zero *)

PROCEDURE LongAdd (A, B : LONG; VAR Result : LONG);
(* Add two LONGs, giving Result *)

PROCEDURE LongSub (A, B : LONG; VAR Result : LONG);
(* Subtract two LONGs (A - B), giving Result *)

PROCEDURE LongToCard (n : CARDINAL; VAR A : LONG);
(* Converts CARDINAL to LONG *)

PROCEDURE LongToCard (A : LONG; VAR n : CARDINAL) : BOOLEAN;
(* Converts LONG to CARDINAL, returns FALSE if conversion impossible *)

PROCEDURE LongToInt (A : LONG; VAR n : INTEGER) : BOOLEAN;
(* Converts LONG to INTEGER, returns FALSE if conversion impossible *)

PROCEDURE LongInc (VAR A : LONG; n : CARDINAL);
(* Increment LONG by n *)

PROCEDURE LongDec (VAR A : LONG; n : CARDINAL);
(* Decrement LONG by n *)

PROCEDURE LongCompare (A, B : LONG) : INTEGER;
(* Returns: 0 if A = B, -1 if A < B, +1 if A > B *)

PROCEDURE LongPut (f : FILE; A : ARRAY OF INTEGER; Size : CARDINAL);
(* Put LONG number in FILE f *)

PROCEDURE LongWrite (A : ARRAY OF INTEGER; Size : CARDINAL);
(* Write LONG number to console screen *)

PROCEDURE StringToLong (S : ARRAY OF CHAR; VAR A : LONG) : BOOLEAN;
(* Converts a string (in HEX) into a LONG *)

PROCEDURE AddrBoundL (VAR A : LONG);
(* Forces Address to a 68000 long word boundary *)

PROCEDURE AddrBoundW (VAR A : LONG);
(* Forces Address to a 68000 word boundary *)

END LongNumbers.
```

End Listing One

LISTING TWO

```
DEFINITION MODULE CmdLin2;
(* Parses command line - returns pointer to an array of pointer to strings *)

FROM SYSTEM IMPORT
ADDRESS;

EXPORT QUALIFIED
ReadCmdLin;

PROCEDURE ReadCmdLin (VAR ArgC : CARDINAL; VAR ArgV : ADDRESS);
(* Gives count of items in command line, and an array of pointer to them *)

END CmdLin2.
```

End Listing Two

LISTING THREE

```
DEFINITION MODULE Parser;
(* Reads the Source file, and splits each *)
(* line into Label, OpCode & Operand(s). *)

FROM Strings IMPORT
STRING;

FROM Files IMPORT
FILE;

EXPORT QUALIFIED
TOKEN, OPERAND, Line, LineCount, OpLoc, SrcLoc, DestLoc, LineParts;

CONST
TokenSize = 8;
OperandSize = 20;

TYPE
TOKEN = ARRAY [0..TokenSize] OF CHAR;
OPERAND = ARRAY [0..OperandSize] OF CHAR;

VAR
OpLoc, SrcLoc, DestLoc : CARDINAL;
Line : STRING;
LineCount : CARDINAL;
```

```
PROCEDURE LineParts (f : FILE; VAR EndFile : BOOLEAN;
VAR Label, OpCode : TOKEN;
VAR SrcOp, DestOp : OPERAND);
(* Reads Line, breaks into tokens, on-passes to symbol & code generators *)

END Parser.
```

End Listing Three

LISTING FOUR

```
DEFINITION MODULE SymbolTable;
(* Initializes symbol table. Maintains list of all labels, *)
(* along with their values. Provides access to the list. *)

FROM LongNumbers IMPORT
LONG;

FROM Parser IMPORT
TOKEN;

EXPORT QUALIFIED
FillSymTab, SortSymTab, ReadSymTab, ListSymTab;

PROCEDURE FillSymTab (Label : TOKEN; Value : LONG; VAR Full : BOOLEAN);
(* Add a symbol to the table *)

PROCEDURE SortSymTab (VAR NumSyms : CARDINAL);
(* Sort symbols into alphabetical order *)

PROCEDURE ReadSymTab (Label : ARRAY OF CHAR;
VAR Value : LONG; VAR Duplicate : BOOLEAN) : BOOLEAN;
(* Passes Value of Label to calling program -- returns FALSE if the *)
(* Label is not defined. Also checks for Multiply Defined Symbols *)

PROCEDURE ListSymTab (i : CARDINAL; VAR Label : TOKEN; VAR Value : LONG);
(* Returns the i-th item in the symbol table *)

END SymbolTable.
```

End Listing Four

LISTING FIVE

```
DEFINITION MODULE CodeGenerator;
(* Uses information supplied by Parser, OperationCodes, *)
(* and SyntaxAnalyzer to produce the object code. *)

FROM Parser IMPORT
TOKEN, OPERAND;

FROM LongNumbers IMPORT
LONG;

EXPORT QUALIFIED
LZero, AddrCnt, Pass2, BuildSymTable, AdvAddrCnt, GetObjectCode;

VAR
LZero, AddrCnt : LONG;
Pass2 : BOOLEAN;

PROCEDURE BuildSymTable (VAR AddrCnt : LONG;
Label, OpCode : TOKEN;
```

End Listing Five

LISTING SIX

```
DEFINITION MODULE SyntaxAnalyzer;
(* Analyzes the operands to provide information for CodeGenerator *)

FROM LongNumbers IMPORT
LONG;

FROM OperationCodes IMPORT
ModeTypeA, ModeTypeB, ModeA, ModeB;

FROM Parser IMPORT
TOKEN, OPERAND, OpLoc, SrcLoc, DestLoc;

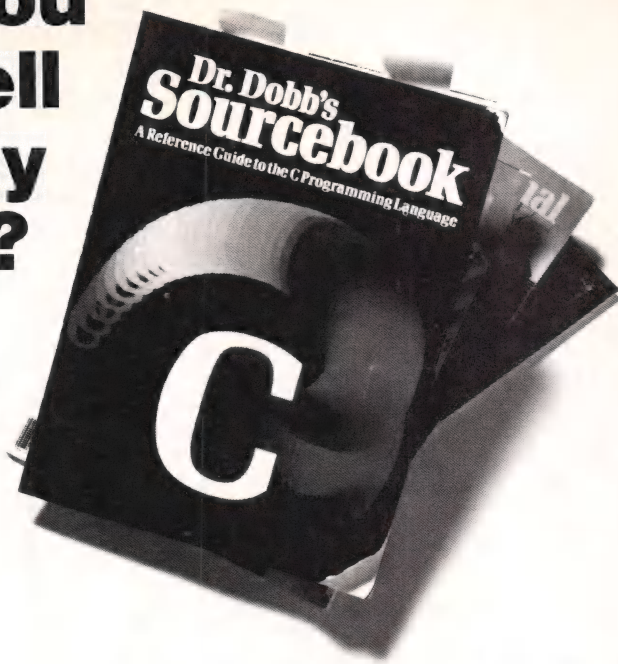
EXPORT QUALIFIED
OpMode, Xtype, SizeType, OpConfig, (* TYPES *)
Size, InstSize, (* VARS *)
AddrModeA, AddrModeB, Op, Src, Dest, (* VARS *)
GetValue, GetSize, (* PROCEDURE's *)
GetInstModeSize, GetOperand, GetMultReg; (* PROCEDURE's *)

TYPE
OpMode = (DReg, (* Data Register *)
ARDir, (* Address Register Direct *)
ARInd, (* Address Register Indirect *)
ARPost, (* Address Register with Post-Increment *)
ARPre, (* Address Register with Pre-Decrement *)
```

(continued on page 78)

Who Says You Can't Tell A Book By Its Cover?

*Dr. Dobb's Sourcebook:
A Reference Guide
for the C Programming
Language*



For years, serious programmers have relied on Dr. Dobb's Journal for the technical tools of their trade. Now, Dr. Dobb's presents the definitive programmers guide to the who, what, where, when and why of C, the leading language among software developers. This comprehensive guide to new information, products and services specific to C will be your most often-used reference!

In this valuable guide you'll find:

- An extensive directory of hardware and software services—including classes and seminars, C programming opportunities, and on-line services
- A bibliography with over 300 listings of available articles and books on C
- A comprehensive C product listing—including C compilers, graphics modules, utilities, editors and development systems, and more!
- And much more practical C programming information

At only \$7.95, no C programmer can afford to be without this unique reference.

ORDER NOW! Call toll-free 1-800-528-6050 Ext. 4001 Ask for item #004

Or, mail this coupon with your payment to: Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063

PAYMENT MUST ACCOMPANY YOUR ORDER

_____ I enclose check/money order

_____ Please charge my _____ VISA _____ M/C _____ American Express

Card # _____ Exp. Date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Please send me _____ copies of **Dr. Dobb's Sourcebook**

at \$7.95 each = _____

+ Shipping & Handling = _____

(Must be included with order. Please add \$1.50 per book in U.S. \$3.25 each surface mail outside U.S. Foreign airmail rates available on request.)

TOTAL = _____

PROMPT DELIVERY!

68K ASSEMBLER

LISTING SIX (Listing continued, text begins on page 52.)

```
ARDisp, (* Address Register with Displacement *)
ARDISX, (* Address Register with Disp. & Index *)
AbsW, (* Absolute Word (16-bit Address) *)
AbsL, (* Absolute Word (32-bit Address) *)
PCDisp, (* Program Counter Relative, with Displacement *)
PCDisX, (* Program Counter Relative, with Disp. & Index *)
Imm, (* Immediate *)
Multim, (* Multiple Register Move *)
SR, (* Status Register *)
CCR, (* Condition Code Register *)
USP, (* User's Stack Pointer *)
Null; (* Error Condition, or Operand missing *)

Xtype = (X0, Dreg, Areg);
SizeType = (S0, Byte, Word, S3, Long);

OpConfig = RECORD (* OPERAND CONFIGURATION *)
    Mode : OpMode;
    Value : LONG;
    Loc : CARDINAL; (* Location of Operand on line *)
    Rn : CARDINAL; (* Register number *)
    Xn : CARDINAL; (* Index Reg. nbr. *)
    Xsize : SizeType; (* size of index *)
    X : Xtype; (* Is index Data or Address register? *)
END;

VAR
    Size : SizeType; (* size for OpCode *)
    AbsSize : SizeType; (* size of operand (Absolute only) *)
    InstSize : CARDINAL;
    AddrModeA : ModeA; (* Addressing modes for this instruction *)
    AddrModeB : ModeB; (* ditto *)
    Op : BITSET; (* Raw bit pattern for OpCode *)
    Src, Dest : OpConfig;

PROCEDURE GetValue (Operand : OPERAND; VAR Value : LONG);
(* determines value of operand (in Decimal, HEX, or via Symbol Table) *)

PROCEDURE GetSize (VAR Symbol : ARRAY OF CHAR; VAR Size : SizeType);
(* determines size of opcode: Byte, Word, or Long *)

PROCEDURE GetAbsSize (VAR Symbol : ARRAY OF CHAR; VAR AbsSize : SizeType);
(* determines size of operand: Word or Long *)

PROCEDURE GetInstModeSize (Mode : OpMode; Size : SizeType;
    VAR InstSize : CARDINAL; CARDINAL);
(* Determines the size for the various instruction modes. *)

PROCEDURE GetOperand (Oper : OPERAND; VAR Op : OpConfig);
(* Finds mode and value for source or destination operand *)

PROCEDURE GetMultReg (Oper : OPERAND; PreDec : BOOLEAN;
    Loc : CARDINAL; VAR MultExt : BITSET);
(* Builds a BITSET marking each register used in a MOVEM instruction *)

END SyntaxAnalyzer.
```

End Listing Six

LISTING SEVEN

```
DEFINITION MODULE ErrorX68;
(* Displays error messages for X68000 cross assembler *)

FROM Files IMPORT
    FILE;

EXPORT QUALIFIED
    ErrorType, ErrorCount, Error, WriteErrorCount;

TYPE
    ErrorType = (Dummy, TooLong, NoCode, SymDup, Undef, SymFull, Phase,
        ModeErr, OperErr, BraErr, AddrErr, SizeErr, EndErr);

VAR
    ErrorCount : CARDINAL;

PROCEDURE Error (Pos : CARDINAL; ErrorNbr : ErrorType);
(* Displays Error #ErrorNbr, then waits for any key to continue *)

PROCEDURE WriteErrorCount (f : FILE);
(* Error count output to Console & Listing file *)

END ErrorX68.
```

End Listing Seven

LISTING EIGHT

```
DEFINITION MODULE Listing;
(* Creates a program listing, including Addresses, Code & Source. *)

FROM Files IMPORT
    FILE;

FROM LongNumbers IMPORT
    LONG;

EXPORT QUALIFIED
    StartListing, WriteListLine, WriteSymTab;
```

```
PROCEDURE StartListing (f : FILE);
(* Sign on messages for listing file -- initialize *)

PROCEDURE WriteListLine (f : FILE;
    AddrCnt, ObjOp, ObjSrc, ObjDest : LONG;
    nA, nO, nS, nD : CARDINAL);
(* Writes one line to the Listing file, including Object Code *)

PROCEDURE WriteSymTab (f : FILE; NumSym : CARDINAL);
(* Lists symbol table in alphabetical order *)

END Listing.
```

End Listing Eight

LISTING NINE

```
DEFINITION MODULE Srecord;
(* Creates Motorola S-records of program: *)
(* S0 = header record, *)
(* S2 = code/data records (24 bit address), *)
(* S8 = termination record (24 bit address). *)

FROM Files IMPORT
    FILE;

FROM LongNumbers IMPORT
    LONG;

EXPORT QUALIFIED
    StartSrec, WriteSrecLine, EndSrec;

PROCEDURE StartSrec (f : FILE; SourceFW : ARRAY OF CHAR);
(* Writes S0 record (HEADER) and initializes *)

PROCEDURE WriteSrecLine (f : FILE;
    AddrCnt, ObjOp, ObjSrc, ObjDest : LONG;
    nA, nO, nS, nD : CARDINAL);
(* Collects Object Code -- Writes an S2 record to file if line is full *)

PROCEDURE EndSrec (f : FILE);
(* Finishes off any left-over (Partial) S2 line, *)
(* and then writes S8 record (TRAILER) *)

END Srecord.
```

End Listing Nine

LISTING TEN

```
MODULE X68000;
(*-----*)
(*
    MC68000 Cross Assembler
    Copyright (c) 1985 by Brian R. Anderson
*)
(*
    This program may be copied for personal, non-commercial use
    only, provided that the above copyright notice is included
    on all copies of the source code. Copying for any other use
    without the consent of the author is prohibited.
*)
(*-----*)

FROM Terminal IMPORT
    WriteString, WriteLn, ReadString;

FROM Files IMPORT
    FILE, FileState, Open, Create, Write, Close;

FROM Strings IMPORT
    STRING, CompareStr, Assign, Concat, Length, Delete;

IMPORT ASCII;

FROM CmdLin2 IMPORT (* Access CP/M command line *)
    ReadCmdLin;

FROM LongNumbers IMPORT
    LONG;

FROM SymbolTable IMPORT
    SortSymTab;

FROM Parser IMPORT
    TOKEN, OPERAND, LineCount, LineParts;

FROM CodeGenerator IMPORT
    LZero, AddrCnt, Pass2, BuildSymTable, AdvAddrCnt, GetObjectCode;

FROM Listing IMPORT
    StartListing, WriteListLine, WriteSymTab;

FROM Srecord IMPORT
    StartSrec, WriteSrecLine, EndSrec;

FROM ErrorX68 IMPORT
    ErrorCount, WriteErrorCount;

TYPE
    FileName = ARRAY [0..14] OF CHAR;
```

(continued on page 80)

PC^{TE}X

Now for PC Users:

Professional Typesetting Capability

PCTeX brings to the personal computer user the ability to put any kind of information on paper in a professional, elegant manner. It brings the full power and flexibility of TeX implementations on mainframes to owners of IBM PC's, AT's and workalike computers.

PCTeX is widely used for formatting technical and mathematical material. It is also perfectly suited for producing professional-quality reports, manuals, even books.

PCTeX offers a wide range of typefaces, and a wide choice of drivers which output the finished material on dot matrix printers (Epson, Toshiba), low-cost laser printers (Apple LaserWriter, Corona LP-300, HP Laser Jet) and graphics screen preview (Hercules, EGA). This ad was formatted by PCTeX and produced on a Corona LP-300.

Join hundreds of satisfied PCTeX users. Write or call us today.

PCTeX: only \$279. Dot-matrix drivers: \$100. Laser drivers: \$300. Preview (Hercules GC): \$250. MF Medley (44 fonts, including Computer Helvetica): \$100. Corona Laser Printer and PCTeX: complete \$3395. System requirements: DOS 2.0 or later, 512K RAM, 10M hard disk. M/C, Visa accepted.

Personal
TeX
Inc

20 Sunnyside, Suite H
Mill Valley, CA 94941
(415) 388-8853 Telex 275611

Trademarks: PCTeX, Personal TeX, Inc.; TeX, American Mathematical Society; IBM PC and AT, IBM Corp; LaserWriter, Apple Computer, Inc.; Hercules Graphics Card, Hercules Computer Technology.

Circle no. 76 on reader service card.

MS-DOS, UNIX, APPLE MAC, CP/M, NETWORKS and MORE. ONE c-tree[®] ISAM DOES THEM ALL!

The creator of Access Manager[™] brings you the most powerful C source code, B+ Tree file handler: **c-tree[™]**

- multi-key ISAM and low-level B+ Tree routines
- complete C source code written to K&R standards
- single-user, network and multi-tasking capabilities
- fixed and variable record length data files
- virtually opened files accommodate limited file descriptors
- no royalties on application programs

\$395 COMPLETE

Specify diskette format:

- 5 1/4" MS-DOS
- 8" CP/M
- 3 1/2" Mac
- 8" RT-II



For VISA, MC and COD orders
call (314) 445-6833

FairCom
2606 Johnson Drive
Columbia, MO 65203

© 1985 FairCom

The following are trademarks: c-tree and the circular disk logo—FairCom; MS—Microsoft Inc.; CP/M and Access Manager—Digital Research Inc.; Unix—AT&T; Apple—Apple Computer Co.

Circle no. 93 on reader service card.

Put More UNIX[™] in Your C.

Unitools \$99

MAKE, DIFF and GREP



These versatile UNIX-style utilities put power at your fingertips. MAKE, a program administrative tool, is like having an assistant programmer at your side. DIFF compares files and shows you the differences between them. GREP can search one or many files looking for one pattern or a host of patterns.



"Z" \$99

A Powerful "vi"-type Editor:

Similar to the Berkeley "vi" editor, "Z's" commands are flexible, terse, and powerful; macro functions give you unlimited range. Features include "undo," sophisticated search and replace functions, automatic indentation, C-tags, and much, much more.



PC-LINT \$99

Error Checking Utility

A LINT-like utility that analyzes programs and uncovers bugs, quirks and inconsistencies. Detects subtle errors. Supports large and small memory models, has clear error messages and executes quickly. Has lots of options and features that you wouldn't expect at this low price.



SunScreen \$99

Low-priced Screen Utility

This versatile graphics package easily creates and modifies formatted screens, validates fields, supports function keys, color and monochrome cards. With library source SunScreen is \$199.

Compatible with all leading MS/PC-DOS C compilers.

SPECIAL OFFER:

Unitools, "Z",
PC-LINT and Sun-
Screen All for only

\$349

To order or for information call:

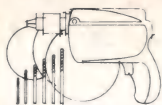
TECWARE
1-800-TEC-WARE

(In NJ call 201-530-6307)



UNIX is a registered TM of Bell Laboratories. MAN'S AZTEC, TM Man Software Systems, Inc. PC-LINT TM GIMPLE software, SunScreen TM Sun Inc. MS-DOS TM Microsoft

Circle no. 222 on reader service card.



Power Tools for system builders™

Call today for our free catalog of design aids, compilers, libraries, debuggers, and support tools for Apple and IBM micro computers. The Power Tools catalog includes product descriptions, warranty and license terms, and all the information you need to make an intelligent purchase decision.

TSF offers technical support, competitive pricing, free UPS shipping on orders over \$100, and a reasonable return policy. Visa, MasterCard, and American Express accepted without surcharge. **TSF helps you get your job done.**

Sample Prices:

Microsoft C \$259
MASM 4.0 \$109
Turbo Pascal \$45
Mark Williams C \$375
Lets C \$59
Wendin OS Toolbox \$89
Blaise Async Manager \$137

Call Toll Free
24 hrs a day/7 days a week

Ask For Operator 2053

800-543-6277

Calif: 800-368-7600



TSF • Dept C-1 • 649 Mission Street
• San Francisco • CA 94105
• (415) 957-0111
The Software Family™

Circle no. 230 on reader service card.

Published for Over 7 Years

68' MICRO JOURNAL

Serving The 68XXX
User Worldwide

The Bible of the
Serious User

\$2.95^{USA}

Australia A \$4.75 New Zealand NZ \$ 6.50
Singapore S \$9.45 Hong Kong H \$20.50
Malaysia M \$9.45 Sweden S 30 SEK

Motorola

68020

68010

68000

68008

6809

Hardware-Software-Application Articles
New Product Releases-Announcements
Hints-Kinks-Fixes, etc.

Subscription Rates

U.S.A.: 1 Yr. \$24.50, 2 Yrs. \$42.50, 3 Yrs. \$64.50

*Foreign Surface: Add \$12.00 per Year to USA Price

*Foreign Airmail: Add \$48.00 per Year to USA Price

*Canada & Mexico: Add \$9.50 per Year to USA Price

* U.S. Currency or Check or Draft Drawn on USA Bank !!

☎ (615) 842-6809



Telex 5106006630



68' Micro Journal a Division of C.P.I.
5900 Cassandra Smith Rd. Hixson Tn. 37343

Circle no. 213 on reader service card.

68K ASSEMBLER

LISTING TEN

(Listing continued, text begins on page 52.)

```
VAR
  ArgC : CARDINAL;
  ArgV : POINTER TO ARRAY [1..3] OF POINTER TO STRING; (* Command Line *)
  SourceFN, ListFN, SrecFN : FileName;
  Source, List, Srec : FILE;
  Label, OpCode : TOKEN;
  SrcOp, DestOp : OPERAND;
  EndFile : BOOLEAN;
  NumSyms : CARDINAL;
  ObjOp, ObjSrc, ObjDest : LONG;
  nA, nO, nS, nD : CARDINAL;
```

```
PROCEDURE MakeNames (VAR S, L, R : FileName);
(* builds names for Source, Listing & S-Record files *)
```

```
VAR
  T : FileName; (* temporary work name *)
  i, l : CARDINAL;

BEGIN
  L := ''; R := ''; (* set Listing & S-rec names to null *)

  i := 0; l := 0;
  WHILE (S[i] # OC) AND (S[i] # ' ') DO
    IF S[i] = '.' THEN (* mark beginning of file extension *)
      l := i;
    END;
    S[i] := CAP (S[i]);
    INC (i);
  END;

  IF S[i] = ' ' THEN
    Delete (S, i, Length (S) - i);
  END;

  Assign (S, T);

  IF l = 0 THEN
    Concat (T, ".ASM", S);
  ELSE
    Delete (T, l, i - l);
  END;

  Concat (T, "_LST", L);
  Concat (T, ".S", R);
END MakeNames;
```

```
PROCEDURE OpenFiles;
BEGIN
  IF Open (Source, SourceFN) # FileOK THEN
    WriteLn;
    WriteString ("No Source File: "); WriteString (SourceFN);
    WriteLn;
    HALT;
  END;

  IF Create (List, ListFN) # FileOK THEN (* DOS may trap this *)
    WriteLn;
    WriteString ("Cannot create disk files!"); WriteLn;
    HALT;
  END;

  IF Create (Srec, SrecFN) # FileOK THEN
    WriteLn;
    WriteString ("Cannot create disk files!"); WriteLn;
    HALT;
  END;
END OpenFiles;
```

```
PROCEDURE StartPass2;
BEGIN
  IF (Close (Source) # FileOK) OR
    (Open (Source, SourceFN) # FileOK) THEN
    WriteString ("Unable to 'Reset' Source file for 2nd Pass.");
    WriteLn;
    HALT;
  END;
  Pass2 := TRUE; (* Pass2 IMPORTed from CodeGenerator *)
  AddrCnt := LZero; (* Assume ORG = 0 to start *)
  ErrorCount := 0; (* ErrorCount IMPORTed from ErrorX68 *)
  LineCount := 0; (* LineCount IMPORTed from Parser *)
  EndFile := FALSE;
END StartPass2;
```

```
PROCEDURE CloseFiles;
BEGIN
  (*-----*)
  (* Ctrl-Z written to files before closing *)
  (* due to bug in "Files" module. Remove these *)
  (* before submitting listing for publication. *)
  (*-----*)
```


Product Information

Free!

Dr. Dobb's Journal of Software Tools

April 1986 #114

Expiration Date: July 30, 1986

Name _____ Title _____

Company _____ Phone _____

Address _____

City/State/Zip _____

Please circle one letter in each category:

I. My work is performed:

- A. for in-house use only.
- B. for other companies.
- C. for end users/retailers.
- D. in none of the above areas.

II. My primary job function:

- A. Software Project Mgmt./Spvr
- B. Hardware Project Mgmt./Spvr
- C. Computer Consultant
- D. Corporate Management
- E. Other

III. My company department performs:

- A. software development.
- B. computer system integration.
- C. computer manufacturing.
- D. computer consulting.
- E. computer research.
- F. none of the above.

IV. This inquiry is for:

- A. a purchase within 1 month.
- B. a purchase within 1 to 6 months.
- C. product information only.

V. Corporate Purchase Authority:

- A. Final Decision-maker
- B. Approve/Recommend
- C. No Influence

VI. Personal Computer Users at my Jobsite:

- A. 10,000 or more
- B. 500 to 9,999
- C. 100 to 499
- D. 10 to 99
- E. less than 10

VII. On average, I advise others about computers:

- A. more than once per day.
- B. once per day.
- C. once per week.
- D. less than once per week.

VIII. In my job function, I:

- A. design software and/or write code.
- B. design software.
- C. write code.
- D. don't design software or write code.

A Reader Service number appears on each advertisement. Circle the corresponding numbers below for more info.

001	002	003	004	005	006	007	008	009
010	011	012	013	014	015	016	017	018
019	020	021	022	023	024	025	026	027
028	029	030	031	032	033	034	035	036
037	038	039	040	041	042	043	044	045
046	047	048	049	050	051	052	053	054
055	056	057	058	059	060	061	062	063
064	065	066	067	068	069	070	071	072
073	074	075	076	077	078	079	080	081
082	083	084	085	086	087	088	089	090
091	092	093	094	095	096	097	098	099
100	101	102	103	104	105	106	107	108
109	110	111	112	113	114	115	116	117
118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153
154	155	156	157	158	159	160	161	162
163	164	165	166	167	168	169	170	171
172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198
199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216
217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234
235	236	237	238	239	240	241	242	243
244	245	246	247	248	249	250	251	252
253	254	255	256	257	258	259	260	261
262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288
289	290	291	292	293	294	295	296	297
298	299	999						

Circle 999 to start a 12 month subscription at the price of \$29.97

Postage Paid!

Thank You!

Dr. Dobb's greatly appreciates your responses to questions I through VIII.

Product Information

Free!

BUSINESS REPLY MAIL

First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of

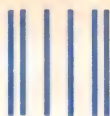
Software Tools

P.O. Box 2157

Clinton, Iowa 52735-2157

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES





NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**Product
Information**

BUSINESS REPLY MAIL

First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of

Software Tools

P.O. Box 2157

Clinton, Iowa 52735-2157



Free!



Thank You!

**Dr. Dobb's greatly appreciates your
responses to questions I through VIII.**

Postage Paid!

A Reader Service number appears on each advertisement. Circle the corresponding numbers below for more info.

001 002 003 004 005 006 007 008 009
010 011 012 013 014 015 016 017 018
019 020 021 022 023 024 025 026 027
028 029 030 031 032 033 034 035 036
037 038 039 040 041 042 043 044 045
046 047 048 049 050 051 052 053 054
055 056 057 058 059 060 061 062 063
064 065 066 067 068 069 070 071 072
073 074 075 076 077 078 079 080 081
082 083 084 085 086 087 088 089 090
091 092 093 094 095 096 097 098 099
100 101 102 103 104 105 106 107 108
109 110 111 112 113 114 115 116 117
118 119 120 121 122 123 124 125 126
127 128 129 130 131 132 133 134 135
136 137 138 139 140 141 142 143 144
145 146 147 148 149 150 151 152 153
154 155 156 157 158 159 160 161 162
163 164 165 166 167 168 169 170 171
172 173 174 175 176 177 178 179 180
181 182 183 184 185 186 187 188 189
190 191 192 193 194 195 196 197 198
199 200 201 202 203 204 205 206 207
208 209 210 211 212 213 214 215 216
217 218 219 220 221 222 223 224 225
226 227 228 229 230 231 232 233 234
235 236 237 238 239 240 241 242 243
244 245 246 247 248 249 250 251 252
253 254 255 256 257 258 259 260 261
262 263 264 265 266 267 268 269 270
271 272 273 274 275 276 277 278 279
280 281 282 283 284 285 286 287 288
289 290 291 292 293 294 295 296 297
298 299 999

Circle 999 to start a 12 month subscription at the price of \$29.97

Dr. Dobb's Journal of Software Tools

April 1986 #114

Expiration Date: July 30, 1986

Name _____ Title _____

Company _____ Phone _____

Address _____

City/State/Zip _____

Please circle one letter in each category:

I. My work is performed:

- A. for in-house use only.
- B. for other companies.
- C. for end users/retailers.
- D. in none of the above areas.

II. My primary job function:

- A. Software Project Mgmt./Spvr
- B. Hardware Project Mgmt./Spvr
- C. Computer Consultant
- D. Corporate Management
- E. Other

III. My company department performs:

- A. software development.
- B. computer system integration.
- C. computer manufacturing.
- D. computer consulting.
- E. computer research.
- F. none of the above.

IV. This inquiry is for:

- A. a purchase within 1 month.
- B. a purchase within 1 to 6 months.
- C. product information only.

V. Corporate Purchase Authority:

- A. Final Decision-maker
- B. Approve/Recommend
- C. No Influence

VI. Personal Computer Users at my Jobsite:

- A. 10,000 or more
- B. 500 to 9,999
- C. 100 to 499
- D. 10 to 99
- E. less than 10

VII. On average, I advise others about computers:

- A. more than once per day.
- B. once per day.
- C. once per week.
- D. less than once per week.

VIII. In my job function, I:

- A. design software and/or write code.
- B. design software.
- C. write code.
- D. don't design software or write code.

**Product
Information**

Free!


```

Write (List, ASCII.sub); Write (Srec, ASCII.sub);

IF (Close (Source) # FileOK)
OR (Close (List) # FileOK)
OR (Close (Srec) # FileOK) THEN
  WriteString ("Error closing files..."); WriteLn;
  HALT;
END;
END CloseFiles;

BEGIN (* X68000 -- main program *)
  ReadCmdLn (ArgC, ArgV);

  IF ArgC = 0 THEN
    WriteLn;
    WriteString ("Enter Source Filename: ");
    ReadString (SourceFN);
    WriteLn;
  ELSE
    Assign (ArgV[1]^, SourceFN);
  END;

  MakeNames (SourceFN, ListFN, SrecFN);

  OpenFiles;

  WriteLn;
  WriteString ("      68000 Cross Assembler"); WriteLn;
  WriteString ("      Copyright (c) 1985 by Brian R. Anderson");
  WriteLn; WriteLn;
  WriteString ("      Assembling "); WriteString (SourceFN);
  WriteLn; WriteLn; WriteLn;

  (*---
  Begin Pass 1
  ---*)
  WriteString ("PASS 1"); WriteLn;
  AddrCnt := LZero; (* Assume ORG = 0 to start *)
  EndFile := FALSE;

  REPEAT
    LineParts (Source, EndFile, Label, OpCode, SrcOp, DestOp);

    BuildSymTable (AddrCnt, Label, OpCode, SrcOp, DestOp);

    AdvAddrCnt (AddrCnt);

  UNTIL EndFile OR (CompareStr (OpCode, "END") = 0);

  (*---
  Begin Pass 2
  ---*)
  WriteString ("PASS 2"); WriteLn;
  StartPass2; (* get Source file, Parser & ErrorX68 ready for 2nd pass *)
  SortSymTab (NumSyms);
  StartListing (List);
  StartSrec (Srec, SourceFN);

  REPEAT
    LineParts (Source, EndFile, Label, OpCode, SrcOp, DestOp);

    GetObjectCode (Label, OpCode,
                   SrcOp, DestOp,
                   AddrCnt, ObjOp, ObjSrc, ObjDest,
                   nA, nO, nS, nD);

    WriteListLine (List, AddrCnt, ObjOp, ObjSrc, ObjDest, nA, nO, nS, nD);

    WriteSrecLine (Srec, AddrCnt, ObjOp, ObjSrc, ObjDest, nA, nO, nS, nD);

    AdvAddrCnt (AddrCnt);

  UNTIL EndFile OR (CompareStr (OpCode, "END") = 0);

  EndSrec (Srec); (* Also: Finish off any partial line *)

  WriteErrorCount (List); (* Error count output to Console & Listing file *)

  WriteSymTab (List, NumSyms); (* Write Symbol Table to Listing File *)

  CloseFiles;

END X68000.

```

End Listings

Now available for the computer experimenter!

COMPUTER CONNOISSEUR'S DELIGHT!

NOW BE IN CONTROL WITH YOUR COMPUTER — THE ONLY PUBLICATION OF ITS KIND WRITTEN FOR THE USER. DISCOVER THE SECRETS AND LEARN THE VERSATILITY OF MODERN COMPUTER COMMAND CONTROL CONCEPTS. EXPERIMENT WITH COMPUTER AND TELEPHONE SYSTEMS, INTERFACE THEM, LEARN HOW THEY WORK, WHAT THEY DO... AND HOW TO GET THEM TO WORK FOR YOU! A COMPLETE TELEPHONE ENGINEERING COURSE IS INCLUDED IN MONTHLY CHAPTERS, BRINGING YOU THROUGH STEP, CROSSBAR, ESS, BUBBLE, AND ATOMIC SWITCHING SYSTEMS! EXCLUSIVE COVERAGE IN BIOLOGICAL COMPUTING SYSTEMS, TOO! COMPUTERS AND TELEPHONES ARE THE FUTURE. THIS PUBLICATION IS AN ABSOLUTE MUST FOR EVERYONE INTERESTED.

UNPUBLISHED
MATERIAL

WIT

COMICS

DIRECTORY
LISTING
NET-
WORKS

AC-
CESS
CODES

*The one you've all
been waiting for*

NOW AVAILABLE — Learn how to repair tele-
phones and telephone systems, how they work, in
monthly installments with the
magazine for you.

Computel™

PUBLISHED MONTHLY

ONE YEAR SUBSCRIPTION \$14.00

(SAMPLE COPY \$2.00)

SUBSCRIPTION & 2 PROGRAMS \$20.00

COMPUTEL—the complete SOURCE for everyone. You can now do the things you've only heard about, right in the privacy of your own home. Indispensable reference to phreaks and hackers. Learn how to get all kinds of computer programs FREE. Get the inside story of big business systems—their quirks and flaws—and remain up to date with vital occurrences within the computer industry. Computel is a publication designed for everyone who has an intense curiosity of computer systems, containing a wealth of hard to find information, codes, and numbers. Published monthly.

Computel Publishing Society

6354 VAN NUYS BL., # 161-B / VAN NUYS, CA 91401

Circle no. 225 on reader service card.



Transform Your Programs with CPP—C Preprocessor Plus

Includes ALL features of the standard C preprocessor.

- Define arbitrarily complex macros with #define command.
- Include and nest files to any depth with #include command.
- Include lines with #if, #ifdef and #ifndef commands.
- Define multiple constants with #enum command.
- Optional extra feature: Imbed formatting or other commands in your source code. (Lines starting with . or * are ignored.)

Fast and flexible

- 30 times faster than the Preprocessor published in Dr. Dobb's Journal.
- Can be used for any language, including assembler.
- Can be used as a stand-alone macro/include processor.
- Code can be used as the lexical analyzer for parsers or assemblers.

Complete

- You get complete SOURCE CODE in standard C.
- You get everything you need to use CPP immediately.
- CPP is unconditionally guaranteed. If for any reason you are not satisfied with CPP, your money will be refunded promptly.

Price: \$95.

Call or write today:
Edward K. Ream
1850 Summit Ave., Dept. DD
Madison, WI 53705
(608) 231-2952

TO ORDER: Specify both the operating system (MS-DOS, CP/M 80 or CPM 68K) and the disk format (8 inch CP/M or the exact type of 5 1/4 inch disk). Send a check or money order for \$95 (\$105 for foreign orders). Foreign checks must be denominated in U.S. dollars drawn on a U.S. bank. Sorry, I do NOT accept phone, credit card or COD orders. Please do NOT send purchase orders unless a check is included.

Circle no. 90 on reader service card.

THE PROGRAMMER'S SHOP™

helps save time, money and cut frustrations. Compare, evaluate, and find products.

SERVICES

- Programmer's Referral List
- Dealer's Inquire
- Compare Products
- Newsletter
- Help find a Publisher
- Rush Order
- Evaluation Literature FREE
- Over 700 products
- BULLETIN BOARD - 7PM to 7AM 617-826-4086

Free Literature - Compare Products

Evaluate products. Compare competitors. Learn about new alternatives. One free call brings information on just about any programming need. Ask for any "Packet" or Addon Packet ☐ AI ☐ ADA, Modula ☐ BASIC ☐ C ☐ COBOL ☐ Editors ☐ FORTH ☐ FORTRAN ☐ PASCAL ☐ UNIX/PC or ☐ Debuggers, Linkers

RECENT DISCOVERIES

ESSENTIAL GRAPHICS - C Library, fast, fonts, no royalties. PC \$250
MULTITASKING - 2 products, available 1 Topview, 1 callable library. Each \$149

AI-Expert System Dev't

Arity System - incorporate with C programs, rule & inheritance PC \$295
1st Class - by example, interfaces \$250
ExpertEASE - Develop by describing examples of how you decide. MS \$595
EXSYS - Improved. Debug. MS \$359
Insight 1 - Probabilities, required thresholds, menus, fast (\$79).
Insight 2 adds backward, forward, partitions, dB2, language, access. MS \$399
Others: APES (\$359), Advisor (\$949), ES Construction (\$100), ESP (\$845), Experteach (\$399), Expert Choice (\$449)

AI-LISP

List Our
GC LISP Interpreter - "Common", rich. Interactive tutorial \$495 Call
GC LISP286 Developer - 2 to 15 meg RAM, compiler & interp. \$1195 Call
Microsoft MuLisp 85 \$250 \$199
TLC LISP - "LISP-Machine" - like, all RAM, classes, compiler. MS \$225
TransLISP - learn fast MS \$ 75
WALTZLISP - "FRANZ LISP" - like, big nums, debug, CPM-80 MS \$149
Others: IQ LISP (\$155), BYSO (\$125)

AI-PROLOG

ARITY Standard - full, 4 Meg Interpreter - debug, C, ASM PC \$ 350
COMPILER/Interpreter-EXE PC \$ 795
With Exp Sys, Screen - KIT PC \$1250
MicroProlog - enhanced MS \$ 229
MProlog - Improved, Faster PC \$ 475
Professional MicroProlog MS \$ 359
Prolog-86 - Learn Fast, Standard, tutorials, samples MS \$ 95
Others: Prolog-I (\$365), Prolog-2 (\$1795)

AI-OTHER

METHODS - SMALLTALK has objects, windows, more PC \$215
QNIAL - Combines APL with LISP. Library of sample programs included. Source or binary. PC \$375
SNOBOL4 + -great for strings, MS \$ 85

FEATURES

Dan Bricklin's Demo Program Prototype quickly. User feedback without programming. All 250 ASC characters plus attributes. Subsetting, macros. PC \$ 75

C Worthy Library - Complete approach, library for applications. Machine independent; network compatible. Source, no royalties, for Lattice MS \$295

BASIC

ACTIVE TRACE, DEBUGGER - BASICA, MBASIC, interactive, well liked MS \$ 79
APC MegaBASIC - powerful PC \$339
BASIC DEVELOPMENT SYSTEM - (BDS) for BASICA; Adds Renum. crossref, compress. PC \$105
Basic Window PC \$ 95
BetterBASIC all RAM, modules, structure. Full BASICA PC \$169
8087 Math Support PC \$ 89
Run-time module PC \$235
CADSAM FILE SYSTEM - full ISAM in MBASIC source. MS \$ 75
CB-86 - DRI CPM86, MS \$449
Data Manager - full source MS \$325
InfoREPORTER - multiple PC \$115
PC/BASIC for Macintosh - by Pteradactyl. Compiles IBM BASICA, and MS BASIC for MAC syntax. \$250
Prof. Basic - Interactive, debug PC \$ 85
8087 Math Support PC \$175
QuickBASIC by Microsoft - Compiles full IBM BASICA, 640K PC \$ 79
TRUE Basic - ANSI PC \$109
Run-time Module PC \$459

COBOL

Macintosh COBOL - full MAC \$459
MBP - Lev II, native MS \$885
MicroFocus Prof. - full PC Call
Microsoft Version II - upgraded. Full Lev. II, native, screens. MS \$500
Realia - very fast MS \$929
Ryan McFarland - portable MS \$699

Editors for Programming

BRIEF Programmer's Editor - undo, windows, reconfigure PC Call
C Screen with source 80/86 \$ 75
EMACS by UniPress - powerful, multifile, windows, DOS, MLISP, programming. Source:\$949 \$299
Entry Systems for C PC \$325
Epsilon - like EMACS, full C-like language. PC \$169
FirstTime by Spruce - Improve productivity. Syntax directed for Turbo (\$69), Pascal (\$229), or C (\$239)
Kedit - like XEDIT PC \$115
Lattice Screen Editor - multiwindow, multitasking Amiga \$100 MS \$125 80/86 \$159
PMATE - power, multitask MS PC \$119
VEDIT - well liked, macros, buffers, CPM-80-86. MS PC \$119
XTC - multitasking PC \$ 85

ATARI ST & AMIGA

We carry full lines of Manx, Lattice, Metacompo & Prospero.

C Language-Compilers

BDS C - solid value, fast CPM80 \$125
C86 by CI - 8087, reliable MS \$299
Consular Mac C w/toolkit MAC \$299
ECO C/88 MS \$ 60
Lattice C - from Lifeboat MS \$289
Lattice C - from Lattice MS \$339
Mark Williams - w/ debugger MS \$399
Megamax - tight full ATARI/ST \$179
Microsoft C 3.0 - new MS \$259
Q/C 88 by Code Works - Compiler source, decent code, cross/native MS \$295
Wizard C - Lattice C compatible, full sys. III, lint, fast. MS \$379

C Language-Interpreters

C-terp by Gimpel - full K & R, .OBJ and ASM, large progs. MS \$249
H.E.L.P. - innovative env. MS \$ 90
INSTANT C - Source debug, Edit to Run-3 seconds MS \$399
Interactive C by IMPACC Assoc. Interpreter, editor, source, debug. PC \$225
Introducing C - Learn C fast, self paced tutorial PC \$109
Professional Run/C - Run/C plus create add-in libraries, load/unload them. MS \$199
Run/C - improved MS \$109

C Libraries-General

Application Programming Toolkit MS \$375
Blaise C Tools 1 (\$109), C Tools 2 \$ 89
C Food by Lattice-ask for source MS \$119
C Utilities by Essential - Comprehensive screen graphics, strings, source. PC \$139
Entelekon C Function Library PC \$119
Entelekon Superfonts for C PC \$ 45
Greenleaf Functions - portable, ASM \$139
Polytron - for Lattice, ASM source \$ 85
Software Horizons - Pack I PC \$129

C Libraries-Communications

Asynch by Blaise \$149
Greenleaf - full, fast \$139
Software Horizons - pack 3 \$119

C Libraries-Files

FILES: C Index by Trio - full B + Tree, vary length field, multi compiler
/File is object only \$ 89
/Pro is partial source \$179
/Plus is full source \$349
C-Tree \$349
CBTREE - multiuser record locking, sequential, source, no royalties \$99

dbVISTA - full indexing, plus optional record types, pointers, Network.
Object only - MS C, LAT, C86 \$179
Source - Single user MS \$459
Source - Multiuser MS \$929

THE PROGRAMMER'S SHOP™

provides complete information, advice, guarantees and every product for Microcomputer Programming.

We support MSDOS (not just compatibles)
PCDOS, Xenix-86, CPM-80, Macintosh,
Atari ST, and Amiga.

SERVICE: FREE NEWSLETTER

Software development and AI on micros: trends, forecasts, controversies, innovations, and techniques. Plus an announcement of 80 NEW tools. CALL for the "Newsletter Packet."

RECENT DISCOVERY

dBrief, the dBASE Assistant - optional syntax directed editing, screen gen, graphics, speed coding. dBASE II, III, Clipper. PC \$ 95

C Support-Systems

Basic C Library by C Source PC \$139
C Debug - Source debuggers - by Complete Soft (\$269), MSD (\$149).
C Sharp - well supported. Source, **realtime**, tasks MS \$600
C ToolSet - DIFF, xref, source MS \$135
Lattice Text Utilities MS \$105
The HAMMER by OES Systems PC \$179
SECURITY LIB - add encrypt to MS C.
C86 programs. Source \$250 PC \$125

C-Screens, Windows, Graphics

Curses by Lattice PC \$109
CView - input, validate PC \$195
C Power Windows by Entelekon PC \$119
Databurst - C or Basic PC \$109
GraphiC - source in C PC \$219
Topview Toolbasket by Lattice PC \$219
View Manager for C by Blaise PC \$219
Windows for C - fast PC \$149
Windows for Data - validation PC \$209

DEBUGGERS

Advanced Trace-86 by Morgan
Modified code on fly. PC \$149
CODESMITH - visual, modify and rewrite Assembler PC \$119
CSPRITE - data structures PC \$149
Periscope I - own 16K PC \$269
Periscope II - symbolic, "Reset Box," 2 Screen PC \$119
Pfix-86 Plus Symbolic Debugger by Phoenix - windows PC \$289
Software Source by Atron - Lattice, MSC, Pascal, Windows single step, 2 screen, log file. MS \$115
w/ Breakswitch PC \$199

Features

Panel Screen Generator - Create screen with editor, generates code. Full data validation, windows, no royalties. Specify Lattice, MSC, C86, MS Fortran or Pascal MS \$239
Microsoft Cobol Tools - symbolic, windowing debugger w/ source support. Plus cross reference, Menu Handler, mouse support. \$210

Fortran & Supporting

Forlib + by Alpha - graphics and file routines, Comm. MS \$ 59
Fortran >> C - FORTRIX C create maintainable **translations**. MS \$5,495
MACFortran by Microsoft - full '77 Includes ASM output MAC \$229
MS Fortran MS \$219
No Limit - Fortran Scientific PC \$129
PolyFortran - xref, pp, screen MS \$149
Prospero - '66, reentrant MS \$390
RM Fortran - enhanced "IBM Professional Fortran" MS \$399
Scientific Subroutines - Matrix MS \$149
Statistician by Alpha MS \$269
Strings and Things - register, shell PC \$ 59

MultiLanguage Support

BTRIEVE ISAM MS \$199
CODESIFTER - Execution PRO-FILER. Spot bottlenecks. Symbolic. automatic. MS \$109
MultiHALO Graphics- Multiple video boards, printer, rich. Animation, engineering, business. Any MS language, Lattice, C86 PC \$189
PLINK 86 - a program-independent overlay linker to 32 levels for all MS languages, C86 and Lattice. MS \$279

Pfinish Performance Analyzer by Phoenix MS \$299
Profiler by DWB Associates MS \$ 99
Screen Sculptor - slick, thorough, fast, BASIC, PASCAL. PC \$109
ZAP Communications - VT 100, TEK 4010 emulation, full xfer. PC \$ 85

TURBO PASCAL and SUPPORT

BORLAND: Turbo 3.0 \$ 49
3.0 with 8087 or BCD \$ 79
3.0 with 8087 and BCD \$ 85
Turbo Graphix - graphs, windows \$ 39
Turbo Toolbox or Editor \$ 55
Turbo Tutor \$ 29
TURBO . . . Asynch by Blaise, full Power Tools by Blaise - library \$ 85
Power Utilities - profiler, pp \$ 85
Professional - interrupts, macros, \$ 50
OTHERS: Screen Sculptor (\$99), Pascal Pac (\$100), Tidy (\$45),

OTHER LANGUAGES

APL + PLUS/PC PC \$469
CLIPPER dBASE Compiler MS \$449
ED/ASM-86 by Oliver Computing. Integrated editor/assembler/debugger w/8087 support. PC \$ 85
HS/FORTH - '79 & '83 Standards, full RAM, ASM, BIOS, interrupts. Graph, multi-task, optimizer MS \$250
MacASM - fast MS \$ 99
MasterForth by MicroMotion - floating point and relocater extensions available: Call MAC or PC \$125
Microsoft MASM - faster MS \$109
Microsoft PASCAL MS \$199
MODULA: M2SDS - popular PC \$ 69
Mystic Pascal - fast PC \$ 64
Paragon PASCAL - for performance: extensions like packages, "Iterators", 5 memory models. 65 bit 8087 strings. Spece vs. speed MS \$665
PASM - by phoenix MS \$219
PL1-86 - Ansi subset PC \$539
Prospero Pascal - full ISO + MS \$390
Turbo Edit/ASM - by Speedware PC \$ 85

XENIX-86 & SUPPORT

Basic - by Microsoft \$295
Cobol - by Microsoft \$895
Fortran - by Microsoft \$429
Xenix Complete Development System \$985

OTHER PRODUCTS

CPRINT - by ENSCO MS \$ 45
dBASE to C Translator: dBx - no royalties, addon ISAM, Library **Pioneer it** MS \$ 350
Source \$1000
HTest/H Format - XT Fix PC \$ 119
Microsoft Windows PC \$ 75
Opt Tech Sort- sort, merge MS \$ 85
Polymake by Polytron MS \$ 85
PS MAKE - Directly execute or Gen a batch file, batch, interactive. MS \$ 129
Qwik Net - critical path, resources, thorough; usable PC \$ 695
SET: SCIL MS \$ 319
SoftEast - Software Estimating and reporting. Pioneer it. MS \$ 350
Texsys - control source MS \$ 89
Visible Computer: 8088 - Simulates demos or any .exe. com, Debugger. 350 pg. tutorial PC \$59

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. All formats available.

Call for a catalog, literature, advice and service you can trust

NEW HOURS

8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP™

128-D Rockland Street, Hanover, MA 02339
Mass: 800-442-8070 or 617-826-7531 286

"I appreciate your service to the programming community, your prices are more than fair, and your newsletter is amongst the finest in the business."

Lawrence T. Fahnoe
Foonan Systems, Ltd.

(continued on next page)

TOTAL RECALL

FOR \$104.95



Now you can retrieve everything you entered during program development -- from version one thru all updates. SRMS does it with a complete source utility at an affordable price.

You can retrieve specific versions of a program, make changes and restate the source while recording when, why and where changes were made.

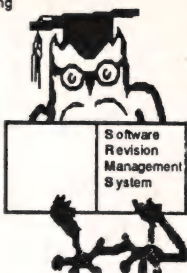
All versions are stored in a single library without duplication of common code or text -- saves disk space. You can also store comments specific to each version -- provides complete development history and documentation of the program.

SRMS supports recall and edit of programs written in BASIC, FORTRAN, PASCAL (including Turbo Pascal), C and ASSEMBLY CODE.

Requires MS or PC DOS, Version 2 and up with 128K and one disk (floppy or hard).

Quilt Computing

7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650



Circle no. 107 on reader service card.

Dr. Dobb's Journal

Subscription Problems?

No Problem!



Give us a call and we'll straighten it out. Today.

Outside California
CALL TOLL FREE: 800-321-3333

Inside California
CALL: 619-485-6535 or 6536

METRIC MINIMIZER

LISTING FOUR (Listing continued)

```

if (gnew >= g)
{
    if (debug>1)
        printf("STOP!!!  new minimum is not lower!\n") ;
    return( BADMIN ) ;
}

/*
 * if the edm (estimated distance to the minimum) is negative we have
 * a catastrophic problem and should stop
 */

matvec( n , y , znewi , temp ) ;
edm = dot( n , znewi , temp ) ;

if( edm < 0.0 )
{
    if (debug>1)
        printf("STOP!!!  edm is negative = %11.4e\n", edm);
    return( NEGEDM ) ;
}

/*
 * now look at normal exits if the minimum number of iterations has
 * been accomplished
 */

if (iterations < itermin) {
    if (debug>1) printf("--->keep going, too few iterations\n") ;
    return( OK ) ;
}

/*
 * edm test: two ways to calculate, stop if either satisfies
 */

if (edm < epsilon[0])
{
    if (debug>1)
        printf("STOP, close enough, edm = %11.4e\n", edm);
    return( EDM1 ) ;
}

edm = dot( n , sigma , sigma ) ;

if (edm < epsilon[0])
{
    if (debug>1)
        printf("STOP, close enough, edm = %11.4e\n", edm);
    return( EDM2 ) ;
}

/*
 * % change in g less than "something" means that approach is too slow
 */

if ( g-gnew < epsilon[1]*g )
{
    if (debug>1)
        printf("STOP, too slow, fractional change = %11.4e\n",
            (g-gnew)/g ) ;
    return( TOOSML ) ;
}

/*
 * fall through case
 * note that case of maximum number of iterations is handled in the
 * calling program
 */

if (debug>1)
    printf("--->keep going, no stoppers found...\n") ;

return( OK ) ;
}
    
```

End Listing Four

LISTING FIVE

```

/* UP.C
 *
 * Routines for updating the matrix y
 * (c) Copyright 1985, Billybob Software. All rights reserved.
 * This program may be reproduced for personal, non-profit use only.
 */

#include "global.h"

/*-----*/

dfpa ( n , sigma , xi , a , debug )
int n ; /* number of parameters */
double *sigma ; /* changes vector */
double *xi ; /* change in gradient vector */
double *a ; /* the result */
int debug ; /* flag = 0 for no print, >0 for debug print */
{
    /*
     * Compute the matrix A which is used to correct Y
     * Davidson Fletcher Powell method
     */

    int i ;
    double norm ;
    double *t ;

    t = a ;
    cross( n , sigma , sigma , a ) ;
    norm = 1.0 / dot( n , sigma , xi ) ;
    i = n*n ;

    while( i-- )
        *a++ *= norm ;

    if (debug>1)
    {
        printf("AAAAA correcting matrix AAAAA is:\n") ;
        mout( n , t ) ;
    }
}

/*-----*/

dfpb ( n , y , xi , b , debug )
int n ; /* number of parameters */
double *y ; /* current Y matrix */
double *xi ; /* change in gradient vector */
double *b ; /* the result */
int debug ; /* flag = 0 if no print, >0 for debug print */
{
    /*
     * compute the matrix B which is used to update Y
     * Davidson Fletcher Powell method
     */

    int i ;
    static double temp[VECMAX] ;
    double *t ;
    double norm ;

    t = b ;
    matvec( n , y , xi , temp ) ;
    norm = - 1.0 / dot( n , temp , xi ) ;
    cross( n , temp , temp , b ) ;
    i = n*n ;

    while( i-- )
        *b++ *= norm ;

    if (debug>1)
    {
        printf("BBBBB correcting matrix BBBBB is:\n") ;
        mout( n , t ) ;
    }
}

```

(continued on next page)

Get the **ProDOS™** advantage for
+
all your **Aztec C65™** programs

VIX is a UNIX like operating system designed to run Manx's Aztec C65 software under ProDOS. With VIX, programs running under the SHELL will run under ProDOS including c65, cci, as65, asi, ln, mklib and others. System includes:

- Standard utilities: cat, cp, date, l, mkdir, ren, rm and stty.
- An improved library written in 6502 assembly.
- A fast screen editor with undelete, auto-indent, work wrap and more.
- Source code to entire system except editor.

VIX - \$49.95 + \$3.50 shipping

Balanced binary tree data base library - \$75
b-tree with source - \$350

Eclipse Systems

223 Matthew Road
Merion Station, Pa. 19066
(215) 664-2419

Circle no. 253 on reader service card.

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)

OUTSIDE OKLAHOMA: NO SALES TAX

V20	\$16.00	V30	\$17.50
8087-2	Math Coprocessors		150.00
DYNAMIC RAM			
256K	64Kx4	150 ns	\$4.75
256K	256Kx1	120 ns	3.37
256K	256Kx1	150 ns	2.79
128K	128Kx1	150 ns	4.75
64K	16Kx4	150 ns	2.50
64K	64Kx1	150 ns	1.35
EPROM			
27C256	32Kx8	250 ns	\$7.35
27128	16Kx8	250 ns	3.15
27C64	8Kx8	250 ns	3.75
2764	8Kx8	250 ns	2.95
STATIC RAM			
6264LP-15	8Kx8	150 ns	\$3.15

640 Kbyte MOTHERBOARD KITS: Zenith 150: \$75.22
IBM PC XT, Compat Portable & Plus: \$70.22

QUANTITY ONE PRICES SHOWN
100/10 256Kx1 D-RAM @ \$5.95

OPEN 6 1/2 DAYS: WE CAN SHIP VIA FED-EX ON SAT.

NO EXTRA COST FOR FED-EX SAT DELIVERY ON ORDERS RECEIVED BY THE 3RD AIR \$5/4 lbs Fr P-One \$13/2 lbs
MasterCard/VISA or UPS CASH COD
Factory New, Prime Parts **µP**
MICROPROCESSORS UNLIMITED, INC.
24,000 S. Peoria Ave., (918) 267-4961
BEGGS, OK, 74421
Prices shown above are for Feb. 24, 1986
Please call for current prices. Prices subject to change. Please expect higher or lower prices on some parts due to supply & demand and our changing costs. Shipping & insurance extra. Cash discount prices shown. Orders received by 6 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air @ \$6.00, or Priority One @ \$13.00.

Circle no. 105 on reader service card.

New Release

CP/M ↔ ISIS
for
PDS & MDS

ICX v.4 eXchanger now supports BOTH 8" MDS and 5-1/4" iPDS formats. Manipulation of ISIS-II files using your CP/M system was never easier.

ISE v.6 Emulator gives the CP/M-80 user access to all the ISIS-II languages and utilities.

Complete source (C and MAC asm) included with all packages

ICXMDS \$89
ICXPDS \$89
ISE \$89
ICX Toolkit (all 3) \$250

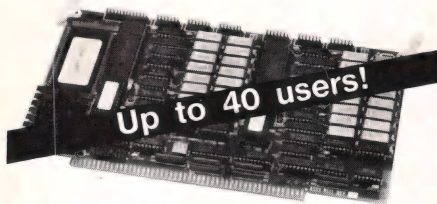
Copyrights: CP/M Digital Research, Inc.
ISIS-II and iPDS Intel Corp.

Western Wares

303-327-4898 • Box C • Norwood, CO 81423

Circle no. 269 on reader service card.

Products With Expandability

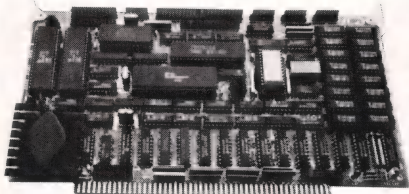


A two user Slave card based on Hitachi's Z80 compatible high speed, 10MHz super microprocessor.

Priced at **\$495⁰⁰***

Features Include...

- 4-10 MHz Z80 Compatible HD64180
- 1/2 Megabyte Nonbanked Memory
- 2 Asynchronous Serial Ports To 38.4
- 1 High Speed Synchronous Port
- All Transfers Via 1.6 MHz DMA!!!
- Unique Expansion Port Offers;
 - 2 Additional Serial Ports or ...
 - 2 Parallel Ports or ...
- Real Time Clock With Battery Backup



The industry's fastest 8-bit Master CPU card with features superior to most 16-bit cards.

Priced at **\$495⁰⁰***

Each Master Features...

- 4-10 MHz Z80 Compatible HD64180
- 1/2 Megabyte Nonbanked Memory
- 2 Asynchronous Serial Ports To 38.4
- 1 High Speed Synchronous Serial Port
- 4 Bi-directional Parallel Ports
- TurboDOS**, ZSYSTEMS**, CP/M**, & OASIS** Operating Systems
- FDC Simultaneously Controls
 - 8", 5 1/4", & 3 1/2" Drives
- SASI/SCSI Interface
- Optional High Speed Hard Disk/File Access Tape Backup and True ETHERNET Controller

*Prices apply to 6 MHz, 64KB versions and are good for a limited time only on purchases of ten or more. For less than ten, please call.

**Trademarks: TurboDOS - Software 2000; ZSYSTEMS - Echelon; CP/M - Digital Research; OASIS - THEOS Software



INTELLIGENT COMPUTER DESIGNS CORP.

23151 Verdugo Drive, Suite 113
Laguna Hills, CA 92653
(714) 581-7500

Circle no. 278 on reader service card.

METRIC MINIMIZER

LISTING FIVE (Listing continued)

```

/*-----*/
bfgsa ( n , y , sigma , xi , a , debug )
int  n ;                /* number of parameters          */
double *y ;             /* current Y matrix            */
double *sigma ;         /* changes vector              */
double *xi ;            /* change in gradient vector   */
double *a ;             /* the result                  */
int  debug ;           /* flag = 0 for no print, >0 for debug print */
{
    /*
     * compute the matrix A which is used to correct Y
     * Broyden Fletcher Goldfarb Shanno method
     */

    int  i , j ;
    static double temp[VECMAX] ;
    static double tmpa[MATMAX] ;
    double norm ;

    norm = 1.0 / dot ( n , sigma , xi ) ;
    matvec( n , y , xi , temp ) ;

    for ( j=0 ; j<n ; ++j )
        temp[j] = sigma[j] - temp[j] ;

    cross( n , temp , sigma , tmpa ) ;
    cross( n , sigma , temp , a ) ;
    i = n*n ;

    for ( j=0 ; j<i ; ++j )
        a[j] = norm * ( a[j] + tmpa[j] ) ;

    if ( debug>1 )
    {
        printf("AAAAA correcting matrix   AAAAA is:\n") ;
        mout( n , a ) ;
    }
}

/*-----*/
bfgsb ( n , y , sigma , xi , b , debug )
int  n ;                /* number of parameters          */
double *y ;             /* current Y matrix            */
double *sigma ;         /* changes vector              */
double *xi ;            /* change in gradient vector   */
double *b ;             /* the result                  */
int  debug ;           /* flag = 0 if no print, >0 for debug print */
{
    /*
     * compute the matrix B which is used to update Y
     * Broyden Fletcher Goldfarb Shanno method
     */

    int  i ;
    static double temp[VECMAX] ;
    double *t ;
    double norm ;

    t = b ;
    norm = 1.0 / dot( n , sigma , xi ) ;
    norm = - norm * norm ;
    matvec( n , y , xi , temp ) ;

    for ( i=0 ; i<n ; ++i )
        temp[i] = sigma[i] - temp[i] ;

    norm *= dot( n , temp , xi ) ;
    cross( n , sigma , sigma , b ) ;
    i = n*n ;

    while( i-- )
        *b++ *= norm ;

```



```

if (debug>1)
{
    printf("BBBBB correcting matrix      BBBBB is:\n") ;
    mout( n , t ) ;
}

/*-----*/

gety ( n , a , b , y , debug )
int    n ;
double *a ;
double *b ;
double *y ;
int    debug ;
{
    /*
     * use matrices A and B to correct Y
     */
    int    i ;
    double *t ;

    t = y ;
    i = n*n ;

    while( i-- )
        *y++ += *a++ + *b++ ;

    if (debug>1)
    {
        printf("YYYYY new matrix YYYYY is:\n") ;
        mout( n , t ) ;
    }
}

```

End Listing Five

LISTING SIX

```

/* UTIL.C
 *
 * Contains miscellaneous routines needed for variable metric minimization
 * program, including matrix and vector utilities
 * (c) Copyright 1985, Billybob Software. All rights reserved.
 * This program may be reproduced for personal, non-profit use only.
 */

#include      "global.h"
#define      INALINE      6

/*-----*/

gozinta( n , xintern , xextern , xlim )
int    n ;
double xintern[] ;
double xextern[] ;
BOUND xlim[] ;
{
    /*
     * transform all external coordinates to internal coordinates
     */

    int    i ;

    for ( i=0 ; i<n ; ++i )
        tointern( i , xintern , xextern , xlim ) ;
}

/*-----*/

gozouta( n , xintern , xextern , xlim )
int    n ;
double xintern[] ;
double xextern[] ;
BOUND xlim[] ;
{
    /*
     * transform all internal coordinates to external coordinates
     */

    int    i ;

```

(continued on next page)

YOU NEED A GOOD LIBRARY



COMPLETE SOURCES NO ROYALTIES

COMPREHENSIVE C Power Packs include over 1000 functions which provide an integrated environment for developing your applications efficiently. "This is a beautifully documented, incredibly comprehensive set of C Function Libraries."

— Dr. Dobb's Journal, July 1984

USEFUL "...can be used as an excellent learning tool for beginning C Programmers..."

— PC User's Group of Colorado, Jan. 1985

FLEXIBLE Most Compilers and all Memory Models supported.

RECOMMENDED "I have no hesitation in recommending it to any programmer interested in producing more applications code, using more of the PC capabilities, in much less time."

— Microsystems, Oct. 1984

- **PACK 1: Building Blocks I** \$149
DOS, Keyboard, File, Printer, Video, Async
- **PACK 2: Database** \$399
B-Tree, Virtual Memory, Lists, Variable Records
- **PACK 3: Communications** \$149
Smartmodem™, Xon/Xoff, X-Modem, Modem-7
- **PACK 4: Building Blocks II** \$149
Dates, Textwindows, Menus, Data Compression, Graphics
- **PACK 5: Mathematics I** \$99
Log, Trig, Random, Std Deviation
- **PACK 6: Utilities I** \$99
(EXE files)
Arc, Diff, Replace, Scan, Wipe

*Master Card/Visa, \$7 Shipping, Mass. Sales Tax 5%

ASK FOR FREE DEMO DISKETTE

**NOVUM
ORGANUM
INC.**

**SOFTWARE
HORIZONS
INC.**

44 Mall Rd., Burlington, MA 01803
(617) 273-4711

Circle no. 262 on reader service card.

METRIC MINIMIZER

LISTING SIX (Listing continued)

```
        for ( i=0 ; i<n ; ++i )
            toextern( i , xintern , xextern , xlim ) ;
    }

/*-----*/

tointern ( j , xintern , xextern , xlim )
int  j ;                /* parameter to transform */
double xintern[] ;      /* internal coordinates */
double xextern[] ;      /* external coordinates */
BOUND xlim[] ;          /* limits on the parameters */
{
    /*
     * transform to unbounded "internal" coordinates used by
     * minimization program.
     */

    double y ;

    if( xlim[j].fl )
    {
        if ( xlim[j].mi == 0.0 )
            xintern[j] = xextern[j] ;
        else
        {
            y = ( xextern[j]-xlim[j].lo ) / xlim[j].mi - 1.0 ;
            xintern[j] = atan( y / sqrt( 1.0 - y*y ) ) ;
        }
    }
    else
        xintern[j] = xextern[j] ;
}

/*-----*/

toextern ( j , xintern , xextern , xlim )
int  j ;                /* parameter to transform */
double xintern[] ;      /* internal coordinates */
double xextern[] ;      /* external coordinates */
BOUND xlim[] ;          /* limits on the parameters */
{
    /*
     * transforms to bounded "external" coordinates known to the real world
     */

    if( xlim[j].fl )
        xextern[j] = xlim[j].lo + xlim[j].mi *
            ( sin(xintern[j]) + 1.0 ) ;
    else
        xextern[j] = xintern[j] ;
}

/*-----*/

derivs ( n , fun , xintern , xstep , xlim , z , data , m , const , debug )
int  n ;                /* number of parameters */
double (*fun)() ;       /* pointer to the function to minimize */
double xintern[] ;      /* coordinates vector */
double xstep[] ;        /* step size to take on each component */
BOUND xlim[] ;          /* limit vector */
double z[] ;            /* derivative vector */
DATA  data[] ;          /* the data */
int  m ;                /* number of data points */
double const[] ;        /* constants required by fcn */
int  debug ;            /* debug flag */
{
    /*
     * compute the derivatives of the vector x using a simple
     * finite difference.
     */

    int  i ;
    static double xextern[VECMAX] ; /* throwaway vector in ext coords */
    static double f1 , f2 ;          /* values of fcn at +/- stepsize */
    static double xtemp ;
```



```

static double eps ;

gozouta( n , xintern , xextern , xlim ) ;
for ( i=0 ; i<n ; ++i )
{
    xtemp      = xintern[i] ;
    eps        = fabs( xtemp ) * xstep[i] ;
    xintern[i] = xtemp + eps ;

    toextern( i , xintern , xextern , xlim ) ;
    f1 = (*fun)( const , xextern , data , m , 0 ) ;
    xintern[i] = xtemp - eps ;

    toextern( i , xintern , xextern , xlim ) ;
    f2 = (*fun)( const , xextern , data , m , 0 ) ;
    z[i] = (f1 - f2) / ( 2.0 * eps ) ;

    if (debug>2)
        printf("i,f1,f2,step,deriv %d %12e %12e %12e %12e\n",
               i, f1, f2, eps , z[i] ) ;

    xintern[i] = xtemp ;
    toextern( i , xintern , xextern , xlim ) ;
}

if (debug==2)
{
    printf("derivatives are:\n");
    vout( n , z ) ;
}

/*-----*/

raz ( nparam , itermin , iterlim , nreset , debug , limit , dstep ,
      param , g0 , epsilon , method )

```

(continued on next page)

FTL Modula-II \$49.95!



Your next computer language. The successor to Pascal, Modula is powerful. Why? Once a routine is written, it need never be recompiled. Programs work everywhere from Z80 through VAX.

FTL Modula-II is a full Z80 CP/M compiler (MSDOS version soon)! It's **fast** -- 18K source compiles in 7 seconds! The built-in split screen editor is worth \$60 alone. Some standard features: full recursion, 15 digit reals, CP/M calls, coprocesses, assembler and linker. The one-pass compiler makes true Z80.COM, ROMable code, too. Get the language you've waited for now. Only \$49.95!

FTL Editor Toolkit

Full source to our split-screen programming editor. Curious? Want to customize to your tastes? Want sample Modula-II code? This is perfect for you. Comes with all you need for your personal editor or terminal installer. Just \$39.95!

Workman and Associates
112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

We have over 200 formats in stock! Please specify your format when ordering. Add \$2.50 per order for shipping. We welcome COD orders!

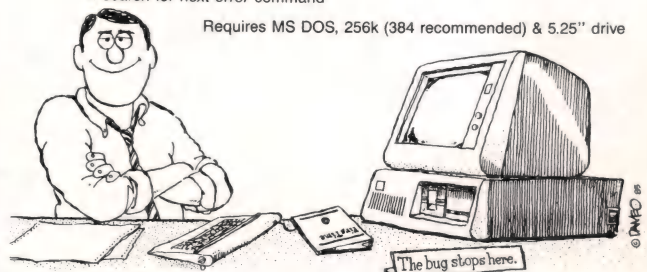
The First Idea-Processor For Programmers.

FirstTime™

Has features no other editor has.

- ☐ Fast program entry through single keystroke statement generators
- ☐ Fast editing through syntax oriented cursor movements
- ☐ Dramatically reduced debugging time through immediate syntax checking.
- ☐ The error checking is thorough and includes semantics
 - Undefined variables, types and constants
 - Assignment statements with mismatched types
 - Errors in include files and macro expansions
- ☐ Automatic program formatter (you specify the rules)
- ☐ Split Screen editing ☐ Command DOS from FirstTime
- ☐ Reading a file with errors moves cursor automatically to point of error
- ☐ Unique programmer-oriented features
 - zoom command gives top-down view of program logic
 - view macro command shows expansion of a C macro in the editor
 - view/update include file allows you to view and update an include file
 - transform command allows you to transform statements to related ones
 - search for next error command

Requires MS DOS, 256k (384 recommended) & 5.25" drive



To Order Call: (201) 741-8188 or write:
SPRUCE TECHNOLOGY CORPORATION



P.O. Box 7948
Shrewsbury, NJ 07701

FirstTime for Turbo Pascal	\$ 74.95
FirstTime for dBase III	\$125.00
FirstTime for MS-Pascal	\$245.00
FirstTime for C	\$295.00

FirstTime is a trademark of Spruce Technology Corporation • MS-DOS is a trademark of Microsoft Corporation • dBase is a trademark of International Business Machines Inc. • Turbo Pascal is a trademark of Borland International • dBase III is a trademark of Ashton-Tate

Circle no. 164 on reader service card.

Circle no. 244 on reader service card.

METRIC MINIMIZER

LISTING SIX (Listing continued)

```

int      *nparam ;           /* number of parameters      */
int      *itermin ;          /* minimum number of iterations */
int      *iterlim ;          /* maximum number of iterations */
int      *nreset ;           /* reset y after nreset iters  */
int      *debug ;            /* debug flag                  */
BOUND    limit[] ;           /* limit vector                */
double    dstep[] ;          /* step size for derivatives    */
double    param[] ;          /* initial values of parameters */
double    *g0 ;              /* expected value of minimum    */
double    epsilon[] ;        /* stopping criteria vector     */
int      *method ;           /* update method for y         */
{
    /*
     * reinitialize important parameters before reading, so that we
     * can later test to see if they were set in the dataset
     */

    int      j ;

    *debug = *nparam = *itermin = *iterlim = *nreset = 0 ;
    *g0 = 0.0 ;
    *method = DFP ;

    for ( j=0 ; j<VECMAX ; ++j )
    {
        limit[j].fl = 0 ;
        dstep[j] = 0.0 ;
        epsilon[j] = 0.0 ;
        param[j] = 0.0 ;
    }
}

/*-----*/

dfault ( n , itermin , iterlim , nreset , epsilon , xlim , dstep , debug )
int      n ;                  /* number of parameters      */
int      *itermin ;           /* minimum number of iterations */
int      *iterlim ;           /* maximum number of iterations */
int      *nreset ;            /* reset y matrix after this many iters */
double    epsilon[] ;         /* stopping criteria vector     */
BOUND    xlim[] ;             /* limit vector for parameters */
double    dstep[] ;           /* step sizes for derivative calc */
int      debug ;              /* flag for debug              */
{
    /*
     * this routine sets defaults if parameters not set with input data
     */

    int      j ;

    if (debug)
    {
        printf("\t\t+++++++\n") ;
        printf("\t\t+ initializations +\n") ;
        printf("\t\t+++++++\n") ;
    }

    /*
     * check that the number of parameters has been set; fatal error if not!
     */

    if ( !n )
        return( -1 ) ;

    /*
     * set up other defaults as required -- these depend on knowing n
     */

    if ( !*itermin )
        *itermin = n ;

    if ( !*iterlim )
        *iterlim = 2*n ;

    if ( !*nreset )

```



```

* nreset = (3*n)/2 ;

for (j=0 ; j<n ; ++j)
    if (dstep[j] == 0.0)
        dstep[j] = 0.01 ;

if (debug)
{
    printf("min iters = %3d, max iters = %3d, ", *itermin ,
           *iterlim ) ;
    printf("reset y every %3d iterations\n", *nreset ) ;
    printf("step sizes for derivatives:\n") ; vout( n , dstep ) ;
}

if ( epsilon[0] == 0.0 )
{
    epsilon[0] = 1.0e-06 ;
    epsilon[1] = 0.001 ;
    if (debug) printf("epsilons set to defaults:\n") ;
}
else if (debug)
    printf("epsilons from input data:\n") ;

if (debug)
    vout( NEPS , epsilon ) ;

/*
 * note that if you add other stopping criteria (more elements in
 * the epsilon vector) you will have to modify this code
 *
 * the following defaults were set in raz and remain if not changed
 * by the data read in reader:
 *   starting values of parameters:      0.0
 *   * expected value of minimum:        0.0
 *   constrained/unconstrained:    unconstrained (all)
 *   parameter names:                blank

```

(continued on next page)

QUICK REF

Indexing at your fingertips! Take the pressure off the tedious search for that much needed article.

Quick Ref offers rapid recall of magazine articles by title, author, and key combinations.

FOR THE INTRODUCTORY PRICE OF

\$34.95

QUICK REF

PROVIDES:

- * Key access to magazine articles
- * Rapid search by title, author, and key combinations
- * Convenient on-line help
- * Easy to use manual

FREE INTRODUCTORY OFFER WITH PURCHASE

The completely indexed
Dr. Dobb's Journal
Data Base

Available for IBM Compatibles and Victor 9000

Terra Base Software
906 S. 8th Street

Laramie, Wyoming 82070

(800) 238-4790 9:00 A.M.—5:00 P.M. M.S.T.

All orders shipped U.P.S. Surface shipping included in price. Visa/MasterCard accepted. Foreign orders please add \$15.00. Checks must be on U.S. Bank in U.S. dollars. Specify machine and DOS version.

Circle no. 231 on reader service card.

FROM THE DEVELOPERS OF THE

65816

Microprocessor

The programming handbook you've been waiting for.

Programming the
65816 Microprocessor
Including 6502 and 65C02



Brady

David Eyes

- * Outlines the programming strengths of the 65816, 6502, and 65C02.
- * Includes a review of basic concepts, architecture, and logical operations.

Now available at your local bookstore, or call toll free 1(800)624-0023 to order your copy today. In New Jersey, call 1(609)624-0024.

0-89303-789-3/288 p/\$22.95

Brady

Circle no. 202 on reader service card.

METRIC MINIMIZER

LISTING SIX (Listing continued)

```

    *      updating method for y :          Davidon-Fletcher-Powell (DFP)
    *
    * variables used for intern<-->extern conversions:
    */

    for (j=0 ; j<n ; j++)
        if (xlim[j].fl)
            xlim[j].mi = (xlim[j].up - xlim[j].lo) / 2.0 ;

    return( 0 ) ;
}

/*-----*/

vout ( n , a )
int    n ;
double *a ;
{
    /*
     * output the floating point vector a with n components
     * INALINE values to a line, indent succeeding lines appropriately
     */

    praline( n , a , 0 ) ;
    putchar('\n') ;
}

/*-----*/

praline( n , a , indent )
int    n ;
double *a ;
int    indent ;
{
    /*
     * print out as many lines as required, indenting as we go
     */

    int    i ;

    if ( indent )
    {
        putchar('\n') ;
        for ( i=indent ; i ; --i )
            putchar(' ');
    }

    for ( i=1 ; i <= n ; ++i )
        printf("%11.4e ", *a++ ) ;

    if ( !n )
        return ;

    praline( n , a , ++indent ) ;
}

/*-----*/

mout ( n , a )
int    n ;
double *a ;
{
    /*
     * output the floating point matrix a with n by n components
     */

    int    i ;
    double *p ;

    for ( i=n , p=a ; i ; --i , p += n )
        vout( n , p ) ;
}

/*-----*/

double dot( n , a , b )

```



```

int    n ;
double *a , *b ;
{
    /*
     * dot product of the vectors a and b, n components each
     */

    double c ;

    c = 0.0 ;
    while( n-- )
        c += *a++ * *b++ ;
    return( c ) ;
}

/*-----*/

reset ( n , y )
int    n ;
double *y ;
{
    /*
     * set the n by n matrix y equal to the identity matrix
     */

    int    i, j;

    for( i = n-1, j = n ; i ; --i, j = n )
    {
        *y++ = 1.0 ;
        while ( j-- )
            *y++ = 0.0 ;
    }
    *y = 1.0 ;
}

/*-----*/

```

(continued on next page)

WORLD'S FIRST ASSEMBLER INTERPRETER

Advanced Trace86™ 2.0

- Create .COM programs with BASIC-like commands: LOAD, SAVE, EDIT, LIST, RUN. Reloadable into AT86, complete with all labels, comments & variable names.
- Macro Assembler support-scan .MAP and .LST files for labels and variable names
- Full-screen symbolic tracing of any program.
- Powerful conditional breakpoint facilities, including hardware "button" support (if installed)
- "Screen save" option, or use dual monitors
- Hex/decimal calculator & converter
- Best 8087/80286/80287 support on the market!
- And more . . . Priced at \$175.00

To order or request more information contact:



Morgan Computing Co., Inc.

(214) 245-4763

P.O. Box 112730, Carrollton, TX 75011

C CODE FOR THE PC

source code, of course

Concurrent C	\$45
Coder's Prolog in C	\$45
LEX	\$25
YACC & PREP	\$25
Small-C compiler for 8088	\$20
tiny-c interpreter & shell	\$20
Xlisp 1.5a & tiny-Prolog	\$20
C Tools	\$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Free shipping on prepaid orders

No credit cards

Circle no. 128 on reader service card.

Circle no. 250 on reader service card.

METRIC MINIMIZER

LISTING SIX (Listing continued)

```

cross( n , v1 , v2 , m )
int    n ;
double *v1 ;
double *v2 ;
double *m ;
{
    /*
     * product of two vectors of dimension n yielding an n by n matrix
     */

    int    i , j ;
    double *p ;

    for( i=n , p=v2 ; i ; --i , ++v1 , p=v2 )
        for( j=n ; j ; --j )
            *m++ = *v1 * *p++ ;
}

/*-----*/

matvec( n , m , v1 , v2 )
int    n ;
double *m ;
double *v1 ;
double *v2 ;
{
    /*
     * product of an n by n matrix and a column vector with n components
     * yielding a second n component vector:  m * v1 = v2
     */

    int    i , j ;
    double *p ;

    for ( i=n , p=v1 ; i ; --i , ++v2 , p=v1 )
        for ( j=n , *v2=0.0 ; j ; --j )
            *v2 += *m++ * *p++ ;
}

```

End Listing Six

LISTING SEVEN

```

/* READ.C
 *
 * reads standard input to get appropriate parameters
 * echoes the data as it is read
 * (c) Copyright 1985, Billybob Software. All rights reserved.
 * This program may be reproduced for personal, non-profit use only.
 */

#include      "global.h"
#define      LEN      130
#define      MAXKEY    19
#define      DEBUGON   if (*debug)
#define      LF        putchar('\n')

/*****/

reader ( fun , const , nparam , param , limit , dstep , pname ,
        epsilon , itermin , iterlim , nreset , g0 , debug ,
        data , npoints , npmax , method )

int    *fun ;           /* address of function (non-portable) */
double const[] ;       /* values of constants used in fcn */
int    *nparam ;       /* number of parameters to be found by fitting */
double param[] ;       /* starting values of parameters */
BOUND limit[] ;        /* on-off flag, upper and lower limits */
double dstep[] ;       /* step sizes used to calculate derivatives */
char    *pname[] ;      /* parameter names */
double epsilon[] ;     /* vector of stopping criteria for iteration */
int    *itermin ;      /* minimum number of iterations required */
int    *iterlim ;      /* maximum number of iterations allowed */
int    *nreset ;       /* # of iters to reset y matrix */
double *g0 ;           /* expected value of minimum */
int    *debug ;        /* debug mode */
DATA   data[] ;        /* data from input file */
int    *npoints ;      /* number of data points */

```

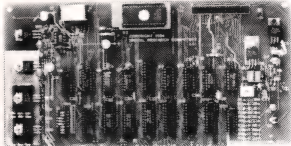
(continued on page 98)

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

\$100 EPROM PROGRAMMER

OUR NEWEST DESIGN, FOR FAST EFFICIENT PROGRAMMING OF THE MOST POPULAR EPROM'S ON YOUR \$100 MACHINE. COMES WITH MENU DRIVEN SOFTWARE THAT RUNS UNDER CP/M 2.2 (8 INCH). PC BOARD SET CONSISTS OF (\$100) MAIN LOGIC BOARD REMOTE PROGRAMMING CARD AND SIX PERSONALITY MINI BOARDS FOR 2716, 2532, 2732, 2732A, 2764, AND 27128. SOLD AS BARE PC BOARD SET ONLY WITH FULL DOC. SOFTWARE FEATURES "FAST" PROGRAMMING ALGORITHM. FOR Z80 BASED SYSTEMS.



PC BOARD SET, FULL DOCUMENTATION, 8 IN. DISKETTE WITH SOFTWARE.

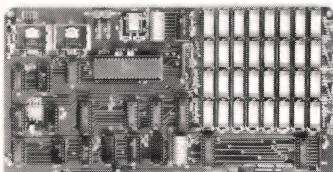
NEW! \$69⁹⁵

128K \$100 STATIC RAM/EPROM BOARD
JUST OUT! USES POPULAR 8K X 8 STATIC RAMS (6264) OR 2764 EPROMS. FOR 8 OR 16 BIT DATA TRANSFERS! IEEE 696 STANDARD. LOW POWER. KITS ARE FULLY SOCKETED. FULL DOC AND SCHEMATICS INCLUDED. 24 BIT ADDRESSING.

NEW! **\$59⁹⁵** **\$219⁰⁰** **\$139⁰⁰**
BARE PC BOARD 128K RAM KIT 128 EPROM KIT

256K S-100 SOLID STATE DISK SIMULATOR!
WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



- FEATURES:**
- 256K on board, using + 5V 64K DRAMS.
 - Uses new Intel 8203-1 LSI Memory Controller.
 - Requires only 4 Dip Switch Selectable I/O Ports.
 - Runs on 8080 or Z80 \$100 machines.
 - Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
 - Provisions for Battery back-up.
 - Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
 - The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
 - Compare our price! You could pay up to 3 times as much for similar boards.

BLANK PCB
(WITH CP/M* 2.2
PATCHES AND INSTALL
PROGRAM ON DISKETTE)
\$49⁹⁵
(8203-1 INTEL \$29.95)

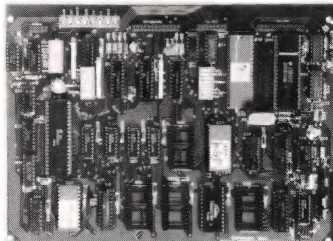
\$129⁰⁰
(ADD \$50 FOR A&T)
#LS-100 (FULL 256K KIT)

ZRT-80

CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

- FEATURES:**
- Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
 - RS232 at 16 BAUD Rates from 75 to 19,200.
 - 24 x 80 standard format (60 Hz).
 - Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
 - Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
 - Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
 - 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
 - Composite or Split Video.
 - Any polarity of video or sync.
 - Inverse Video Capability.
 - Small Size: 6.5 x 9 inches.
 - Upper & lower case with descenders.
 - 7 x 9 Character Matrix.
 - Requires Par. ASCII keyboard.



\$89⁹⁵ **A&T**
#ZRT-80 **ADD \$50**
(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716
CHAR. ROM. 2732 MON. ROM
\$49⁹⁵
SOURCE DISKETTE - ADD \$10
SET OF 2 CRYSTALS - ADD \$7.50

FOR 8 IN. SOURCE DISK
(CP/M COMPATIBLE)
ADD \$10

64K \$100 STATIC RAM

\$99⁰⁰
KIT

LOW POWER!
150 NS ADD \$10

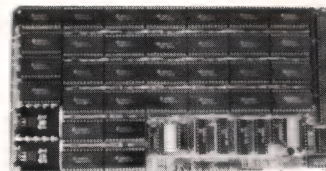
BLANK PC BOARD
WITH DOCUMENTATION
\$49.95

SUPPORT ICs + CAPS
\$17.50

FULL SOCKET SET
\$14.50

FULLY SUPPORTS THE
NEW IEEE 696 \$100
STANDARD
(AS PROPOSED)

ASSEMBLED AND
TESTED ADD \$50



- FEATURES: PRICE CUT!**
- Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
 - Fully supports IEEE 696 24 BIT Extended Addressing.
 - 64K draws only approximately 500 MA.
 - 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
 - SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
 - 2716 EPROMs may be installed in any of top 48K.
 - Any of the top 6K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
 - Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
 - BOARD may be partially populated as 56K.

PANASONIC

Green Screen - Video Monitors

25 MHZ. TYPICAL BANDWIDTH!!!

Brand New In The Box! 9-Inch Screen

#K-904B1 (Chassis #Y08A) Open Frame Style

\$29⁹⁵
EA.

GROUP SPECIAL:
4 for \$99⁰⁰
WITH DATA & SCHEMATIC

(USA SHIPPING: \$3. PER UNIT. CANADA: \$7. PER UNIT)

COMPUTER MANUFACTURER'S EXCESS. STILL IN ORIGINAL PANASONIC BOXES. THE CRT TUBE ALONE WOULD COST MORE THAN OUR PRICE FOR THE COMPLETE UNIT. FOR SPLIT VIDEO (TTL INPUTS) OPERATION, NOT COMPOSITE VIDEO. OPERATES FROM 12VDC AT 1 AMP. VERTICAL INPUT IS 49 TO 61 HZ. HORIZONTAL INPUT: 15,750 HZ ± 500 HZ. RESOLUTION IS 800 LINES AT CENTER 650 LINES AT CORNERS.

THE NEW 65/9028 VT ANSI VIDEO TERMINAL BOARD!

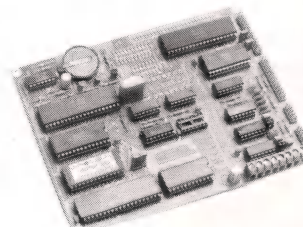
★ FROM LINGER ENTERPRISES ★

A second generation, low cost, high performance, mini sized, single board for making your own RS232 Video Terminal. Use as a computer console or with a MODEM for hook up to any of the telephone-line computer services.

FEATURES:

- Uses the new SMC 9028 Video Controller Chip coupled with a 6502A CPU.
- RS-232 at 16 Baud Rates from 50 to 19,200.
- On board printer port!
- 24 X 80 format (50/60 Hz).
- For 15,750 Hz (Horiz.) monitors.
- 3 Terminal Modes: H-19, ADM3A, and ANSI X 3.64-1979
- Wide and thin-line graphics.
- White characters on black background or reversed.
- Character Attributes: De-Inten, Inverse or Underline.
- Low Power: 5VDC @ .7A, ± 12VDC @ 20MA.
- Mini size: 6.5 X 5 inches.
- Composite or split video.
- 5 X 8 Dot Matrix characters (U/L case).
- Answer back capability.
- Battery backed up status memory.
- For ASCII parallel keyboard.

MICRO SIZE!



\$99⁹⁵
(Full Kit)

SOURCE DISKETTE:
PC/XT FORMAT
5 1/4 IN. \$15

ADD \$40 FOR A&T

TERMS: Add \$3.00 postage. Orders under \$15 add 75¢ handling. No C.O.D. We accept Visa and MasterCard. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50 add 85¢ for insurance.

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

METRIC MINIMIZER

LISTING SEVEN (Listing continued)

```

int    npmax ;          /* maximum number of data points allowed */
int    *method ;        /* dfp or bfgs updating method */
{
    int    nconst ;      /* number of constants used in fcn */
    char    instring[LEN] ;
    char    *restofline ;
    char    *firstword ;
    static char    funname[] = "          " ;
    static char    *key[MAXKEY] =
    {
        "bfgs" ,          /*      0      */
        "constants" ,     /*      1      */
        "debug" ,         /*      2      */
        "derivsteps" ,    /*      3      */
        "dfp" ,           /*      4      */
        "end" ,           /*      5      */
        "epsilons" ,      /*      6      */
        "exmin" ,         /*      7      */
        "funname" ,       /*      8      */
        "iterlim" ,       /*      9      */
        "itermin" ,      /*     10      */
        "limitflags" ,    /*     11      */
        "lowerlimits" ,   /*     12      */
        "newdata" ,       /*     13      */
        "next" ,          /*     14      */
        "pstart" ,       /*     15      */
        "reset" ,        /*     16      */
        "sd" ,           /*     17      */
        "upperlimits"    /*     18      */
    } ;

    /*
     * note above order; keywords are tested below in same order...
     * remember this when adding keywords!
     */

    int    i , j , ncmd , ncom , bad ;

#ifdef C80
    int chan ;
#else
    FILE    *chan ;
    FILE    *fopen() ;
    char    *fgets() ;
#endif

    /*
     * loop until a "next" or "end" command is picked up
     */

    ncmd = ncom = bad = 0 ;
    for (;;)
    {
#ifdef C80
        if ( ! getline( instring , LEN ) )
            return( ALLDONE ) ;

        printf("%s\n", instring) ;

        /*
         * C-80 library routine getline returns the line stored into
         * "instring" with the null at the end, but with the '\n'
         * stripped off.
         */

#else
        if ( fgets( instring , LEN , stdin ) == NULL )
            return( ALLDONE ) ;

        printf("%s", instring) ;

#endif

        restofline = instring ;
        getnext( &restofline , &firstword ) ;
    }
}

```



```

for ( j=0 ; j<MAXKEY ; ++j )
    if ( ! strcmp( firstword , key[j] ) )
        break ;

switch (j)
{
/* bfgs */
case 0:
    ++ncmd ;
    DEBUGON printf("bfgs update requested\n") ;
    *method = BFGS ;
    break ;

/* constants */
case 1:
    ++ncmd ;
    DEBUGON printf("%d constants in %s:\n", nconst ,
                    funname ) ;
    for (i=0 ; i < nconst ; ++i)
    {
        getnext( &restofline , &firstword ) ;
        if( check( firstword ) )
        {
            ++bad;
            break;
        }
        const[i] = (double) atof( firstword ) ;
        DEBUGON printf("%e ", const[i]);
    }
    DEBUGON LF;
    break ;

/* debug */
case 2:
    ++ncmd ;
    getnext( &restofline , &firstword ) ;
    if ( check( firstword ) )
    {

```

(continued on next page)

DE SMET C

8086/8088 Development Package

\$109

FULL DEVELOPMENT PACKAGE

- Full K&R C Compiler
- Assembler, Linker & Librarian
- Full Screen Editor
- Execution Profiler
- Complete STDIO Library (>120 Func)

Automatic DOS 1.X/2.X SUPPORT

BOTH 8087 & S/W FLOATING POINT

OVERLAYS

OUTSTANDING PERFORMANCE

- First and Second in AUG '83 BYTE benchmarks

SYMBOLIC DEBUGGER \$50

- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

DOS LINK SUPPORT \$35

- Converts DeSmet.O to DOS.OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

CWARE
CORPORATION

P.O. Box C, Sunnyvale, CA 94087
(408) 720-9696

Street Address: 505 W. Olive, #767 (94086) Call for hrs.

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add \$5, elsewhere add \$15. Checks must be on U.S. Bank and in U.S. Dollars. Call 9am-1pm to CHARGE by VISA/MC/AMEX.

Foreign Distributors: **AFRICA**, HI-TECH SVCS, Gaborone 4540 or Telex 22058D LANGER • **ENGLAND**: MLH Tech, 0606-891146 • **JAPAN**: JSE 03-486-7151 • **SWEDEN**: ESCORT DATA 08-87 41 48 or THESEUS KONSULT 08-23 61 60

BLAST®



PC-MINI-MAINFRAME COMMUNICATIONS SOFTWARE

ANY COMPUTER WITH BLAST CAN TALK TO ANY OTHER COMPUTER WITH BLAST, the universal file transfer utility linking many different computers, operating systems, and networks, via RS 232 serial ports.

NO ADD-ON BOARDS TO BUY! BLAST software uses any asynchronous modems or direct connect for fast, error-free data transfer through noisy lines and PBXs, across LANs, and over satellites or packet switched networks.

THE PERFECT LOW-COST LINK FOR PC's, MINIS, MAINFRAMES Transfer binary or text files, or executable commands. Use BLAST standalone, or built it into your application.

\$250/Micros \$495-895/Minis \$2495/up Mainframes

COMMUNICATIONS RESEARCH GROUP (800)-24-BLAST

8939 Jefferson Hwy. Baton Rouge, LA 70809 (504)-923-0888

Circle no. 272 on reader service card.

METRIC MINIMIZER

LISTING SEVEN (Listing continued)

```

        ++bad;
        break;
    }

    *debug = atoi( firstword );
    DEBUGON printf("debug level is %d\n", *debug);
    break ;

/* derivsteps */
case 3:
    ++ncmd ;
    DEBUGON printf("derivsteps are:\n");
    for (i=0 ; i < *nparam ; ++i)
    {
        getnext( &restofline , &firstword ) ;
        if ( check( firstword ) )
        {
            ++bad;
            break;
        }
        dstep[i] = (double) atof( firstword ) ;
        DEBUGON printf("%e ", dstep[i]);
    }
    DEBUGON LF;
    break ;

/* dfp */
case 4:
    ++ncmd ;
    DEBUGON printf("dfp update requested\n") ;
    *method = DFP ;
    break ;

/* end */
case 5:
    DEBUGON printf("end of all data\n") ;
    return( ALLDONE ) ;

/* epsilons */
case 6:
    ++ncmd ;
    DEBUGON printf("%d epsilons for cutoffs:\n",
        NEPS );
    for (i=0 ; i < NEPS ; ++i)
    {
        getnext( &restofline , &firstword ) ;
        if( check( firstword ) )
        {
            ++bad;
            break;
        }
        epsilon[i] = (double) atof( firstword );
        DEBUGON printf("%e ", epsilon[i]);
    }
    DEBUGON LF;
    break ;

/* exmin */
case 7:
    ++ncmd ;
    getnext( &restofline , &firstword ) ;
    if ( check( firstword ) )
    {
        ++bad;
        break;
    }
    *g0 = (double) atof( firstword ) ;
    DEBUGON printf("expected minimum is %11.4e\n", *g0);
    break ;

/* funname */
case 8:
    ++ncmd ;
    getnext( &restofline , &firstword ) ;
    if( check( firstword ) )
    {
        ++bad;
        break;
    }
    strcpy( funname , firstword ) ;
    DEBUGON printf("function to minimize is %s\n",
        funname ) ;

```

(continued on page 102)

Now You Know Why **BRIEF** is **BEST**

"BRIEF, The Programmer's Editor, is simply the best text editor you can buy." John Dvorak, INFOWORLD 7/8/85

The Program Editor with the **BEST** Features

Since its introduction, BRIEF has been sweeping programmers off their feet. Why? Because BRIEF offers the features **MOST ASKED FOR** by professional programmers. In fact, BRIEF has just about every feature you've ever seen or imagined, including the ability to configure windows, keyboard assignments, and commands to **YOUR** preference. One reviewer (David Irwin, DATA BASED ADVISOR) put it most aptly, "(BRIEF)... is quite simply the best code editor I have seen."

Solution Systems™

MACRO LANGUAGE

As Steve McMahon describes in Byte (3/85), "BRIEF is even more customizable than [another customizable editor] - not only may wholly new commands be created, but they may be assigned to any key, replacing even basic function keys like cursor control keys or the return key... String and integer [variable] types are available and may have either global or local scope... Arithmetic and logical primitives, type conversion, buffer and window control, search and translate, keyboard input, ... are [also] available. "Much of BRIEF was written in the BRIEF macro language, and the source of these macros is included with the editor. Using this source, it's possible to customize even sophisticated functions of the editor..."

No other editor has a more powerful macro language.

Every Feature You Can Imagine

Compare these features with your editor (or any other for that matter).

- FAST
- Full UNDO (N Times)
- Edit Multiple Large Files
- Compiler-specific support, like auto indent, syntax check, compile within BRIEF, and template editing
- Exit to DOS inside BRIEF
- Uses all Available Memory
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning Time
- Windows (Tiled and Pop-up)
- Unlimited File Size - (even 2 Meg!)
- Reconfigurable Keyboard
- Context Sensitive Help
- Search for "regular expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- A Complete Compiled Programmable and Readable Macro Language
- EGA and Large Display Support
- Adjustable line length up to 512

Program Editing **YOUR** Way

A typical program editor requires you to adjust your style of programming to its particular requirements - NOT SO WITH BRIEF. You can easily customize BRIEF to your way of doing things, making it a natural extension of your mind. For example, you can create ANY command and assign it to ANY key - even basic function keys such as cursor-control keys or the return key.

The Experts Agree

Reviewers at BYTE, INFOWORLD, DATA BASED ADVISOR, and DR. DOBB'S JOURNAL all came to the same conclusion - **BRIEF IS BEST!**

Further, of 20 top industry experts who were given BRIEF to test, 15 were so impressed they scrapped their existing editors!

NOT COPY PROTECTED

MONEY-BACK GUARANTEE

Try BRIEF (\$195) for 30 days - If not satisfied get a full refund.
TO ORDER CALL (800-821-2492)

METRIC MINIMIZER

LISTING SEVEN (Listing continued)

```
        if ( funlib( funname, fun, nparam, &nconst, pname))
        {
            DEBUGON printf("function not in library!\n") ;
            ++bad ;
        }
        break ;

/* iterlim */
case 9:
    ++ncmd ;
    getnext( &restofline , &firstword ) ;
    if( check( firstword ))
    {
        ++bad;
        break;
    }
    *iterlim = atoi( firstword ) ;
    DEBUGON printf("iterlim is %d\n", *iterlim);
    break ;

/* itermin */
case 10:
    ++ncmd ;
    getnext( &restofline , &firstword ) ;
    if( check( firstword ))
    {
        ++bad;
        break;
    }
    *itermin = atoi( firstword ) ;
    DEBUGON printf("itermin is %d\n", *itermin);
    break ;

/* limitflags */
case 11:
    ++ncmd ;
    DEBUGON printf("limitflags are:\n") ;
    for (i=0 ; i < *nparam ; ++i)
    {
        getnext( &restofline , &firstword ) ;
        if ( check( firstword ))
        {
            ++bad;
            break;
        }
        limit[i].fl = atoi( firstword ) ;
        DEBUGON printf("%d ", limit[i].fl);
    }
    DEBUGON LF;
    break ;

/* lowerlimits */
case 12:
    ++ncmd ;
    DEBUGON printf("lowerlimits are:\n") ;
    for (i=0 ; i < *nparam ; ++i)
    {
        getnext( &restofline , &firstword ) ;
        if ( check( firstword ))
        {
            ++bad;
            break;
        }
        limit[i].lo = (double) atof( firstword ) ;
        DEBUGON printf("%e ", limit[i].lo);
    }
    DEBUGON LF;
    break ;

/* newdata */
case 13:
    ++ncmd ;
    getnext( &restofline , &firstword ) ;
    DEBUGON printf("datafile requested was %s\n",
                    firstword ) ;
    if ( check( firstword ))
    {
        ++bad;
        break;
    }
```



```

    }

    chan = fopen( firstword , "r" );

    if ( !chan )
    {
        printf("datafile can't be opened!\n") ;
        return( ALLDONE ) ;
    }

    fscanf( chan , "%d" , npoints ) ;
    DEBUGON printf("%d datapoints in file\n", *npoints);

    if (*npoints > npmax)
    {
        ++bad ;
        printf("more data than allowed!\n") ;
        printf("%d datapoints in file\n", *npoints);
        printf("%d points allowed for\n", npmax ) ;
        break ;
    }

    for (i=0 ; i<*npoints ; ++i)
    {
#ifdef C80
        fscanf(chan, "%f %f", &data[i].x, &data[i].y);
#else
        fscanf(chan,"%lf %lf",&data[i].x, &data[i].y);
#endif
        DEBUGON printf("%3d %f %f\n" , i+1 ,
            data[i].x , data[i].y ) ;
    }
    fclose( chan ) ;
    break ;

/* next      */
case 14:
    ++ncmd ;

```

(continued on next page)

PC/VI

Full Screen Editor for MS-DOS (PC-DOS)

Looking for an Ultra-Powerful Full-Screen editor for your MS-DOS or PC-DOS system? Are you looking for an editor FULLY COMPATIBLE with the UNIX* VI editor. Are you looking for an editor which not only runs on IBM-PC's and compatibles, but ANY MS-DOS system? Are you looking for an editor which provides power and flexibility for both programming and text editing? If you are, then look no further because PC/VI IS HERE!

The following is only a hint of the power behind PC/VI: English-like syntax is command mode, mnemonic control sequences in visual mode; full undo capability; deletions, changes and cursor positioning on character, word, line, sentence, paragraph or global basis; editing of files larger than available memory; powerful pattern matching capability for searches and substitutions; location marking; joining multiple lines; auto-indentation; word abbreviations and MUCH, MUCH MORE!

The PC/VI editor is available for IBM-PC's and generic MS-DOS based systems for only \$149. For more information call or write:

Custom Software Systems
P.O. Box 551 MO
Shrewsbury, MA 01545
617-842-1712

The UNIX community has been using the VI editor for years. Now you can run an implementation of the same editor under MS-DOS. Don't miss out on the power of PC/VI!

*UNIX is a trademark of AT&T Bell Laboratories.

Circle no. 268 on reader service card.

FORTRAN PROGRAMMERS

Downloading from mainframes or developing on the PC the choice is F77L.

"Lahey's F77L FORTRAN is the compiler of choice... F77L compiled the five files in a total of 12 minutes which was 4 times as fast as MS FORTRAN and an astounding 6 times as fast as Pro FORTRAN."

PC Magazine

"The manual that comes with this compiler is well put together. The messages are clearly explained, the compiler's unique features are well documented... All in all, F77L is a fine, well supported product that we think will do very well in the marketplace."

Computer Language

VERSION 2.0 NOW AVAILABLE — \$477

Full ANSI FORTRAN-77
Source On-Line Debugger
Common/Array greater than 64K
Lattice C and other 3rd Party Compatibility

To order or for more information

(213) 541-1200

Lahey Computer Systems, Inc.
31244 Palos Verdes Drive West, Suite #243
Rancho Palos Verdes, CA 90274

Requires MS-DOS and 8087

MS-DOS and MS FORTRAN are trademarks of Microsoft Corporation
Pro FORTRAN is a trademark of International Business Machines

Circle no. 186 on reader service card.

METRIC MINIMIZER

LISTING SEVEN (Listing continued)

```

        DEBUGON printf("end of this data set\n");
        printf("%2d command lines read\n", ncmd) ;
        printf("%2d comment lines read\n", ncom) ;

        if( ! *nparam )
            ++bad ;

        return( bad ) ;

/* pstart */
case 15:
    ++ncmd ;
    DEBUGON printf("starting values are:\n" ) ;
    for (i=0 ; i < *nparam ; ++i)
    {
        getnext( &restofline , &firstword ) ;
        if ( check( firstword ) )
        {
            ++bad;
            break;
        }
        param[i] = (double) atof( firstword ) ;
        DEBUGON printf("%e ", param[i]);
    }
    DEBUGON LF;
    break ;

/* reset */
case 16:
    ++ncmd ;
    getnext( &restofline , &firstword ) ;
    if( check( firstword ) )
    {
        ++bad;
        break;
    }
    *nreset = atoi( firstword ) ;
    DEBUGON printf("reset is %d\n", *nreset);
    break ;

/* sd */
case 17:
    ++ncmd ;
    DEBUGON printf("steepest descent update requested\n") ;
    *method = STDES ;
    break ;

/* upperlimits */
case 18:
    ++ncmd ;
    DEBUGON printf("upperlimits are:\n");
    for (i=0 ; i < *nparam ; ++i)
    {
        getnext( &restofline , &firstword ) ;
        if ( check( firstword ) )
        {
            ++bad;
            break;
        }
        limit[i].up = (double) atof( firstword ) ;
        DEBUGON printf("%e ", limit[i].up);
    }
    DEBUGON LF;
    break ;

/*
 * anything else is a comment; throw away and go
 * to next line.
 */

default:
    ++ncom ;
    DEBUGON printf("***%s*** taken as comment\n",
                    firstword ) ;
    }
}

/*****/

```

(continued on page 106)

C Programmers:

Here are 8 ways You can be more productive

Dear Microcomputer Programmer,

Let me tell you how you can find and choose the best development software for your needs — software that will help you:

- * Speed your development efforts
- * Write even better programs
- * Increase productivity
- * Reduce your programming frustration

All you have to do is consider one of these eight products for C programmers (or the 97 other C compilers, interpreters, support libraries, debuggers, or addons we offer). Then call one of our knowledgeable consultants - toll free - for details, comparisons, or for our specially prepared packets on C, C Libraries, or C Productivity Tools.

There is no obligation. You risk nothing with our moneyback guarantee of satisfaction.

Yours for more productive programming, — Bruce W. Lynch, President

First Aid for C Programs C ToolSet

Save time and frustration when analyzing and manipulating C programs.

DIFF and CMP-for "intelligent" file comparisons.

XREF - cross references variables by function and line.

C Flow chart - shows what functions call each other.

C Beautifier - make source more readable.

GREP - search for patterns.

PP - formats your code so that it is easier to read and understand.

C Util - acts as a general purpose file filter.

There are several other programs for converting and printing programs.

Portable. Full source code.

CPM, MSDOS \$135

Even for Small Files: Convenient, Fast Access

CBTREE — Only \$99

Why spend time writing file management code when you can use consistent, flexible, documented, professional function? Even multiuser record locking and variable-length records are supported.

Full, balanced Btree support includes use of multiple keys, unlimited number and length of keys.

Use this powerful ISAM, even if you've previously done without.

Learn how to write systems for managing large files by using CBTREE source as a guide. Modify it and transfer it to another operating environment without royalties.

SORT/MERGE Files for Clean, Fast Maintenance

with OPT-TECH SORT

Performance should not suffer with DOS or other "free" sorts. ISAMs alone are slow when 10% or even less is changed/added. OPT-TECH includes:

- CALLable and Standalone use
- C, ASM, BAS, PAS, FTN, COBOL
- Variable and fixed length
- 1 to 9 fields to sort/merge
- Autoselect of RAM or disk
- Options: dBASE, Btrieve files
- 1 to 10 files input
- No software max for # Records
- All common field types
- By pass headers, limit sort
- Inplace sort option
- Output = Record or keys

Try what you're using on an XT: 1,000 128 byte records, 10 byte key in 33 seconds. MSDOS \$85

Add Communications Features to Your Programs

Greenleaf Comm Library

Greenleaf now enables you to communicate with remote systems or databases with an asynchronous communications library for C.

Individual transmission and reception ring buffers combine with an interrupt driven system. This eliminates the extra function of separately calling up the communications program.

Included are 1 library/object files, 100 functions; 100 page manual, complete source code, library tailor-made to suit compiler and memory. Hayes-compatible modem commands, and a complete sample file transfer program. MSDOS \$149

Get File Access with TIGHTER Control

db_VISTA Data Management

Full source, no royalties and "normal" indexed file management are part of db_VISTA. Get more for the price of only an ISAM.

You can minimize data stored and access records even faster and more logically than just using indexes. Example: address and transaction data should not require redundant storage of customer names or numbers. Use pointers. Related data fields point to other related groups - the "network model" of data.

Use db_VISTA as a "normal ISAM" or save programming time, access time and file size. Lattice, C86, Williams, Desmet, Microsoft C.

MSDOS Multiuser source \$995, Object \$495

Single user source \$450, Object \$169

Unix, Xenix, & MacIntosh versions also available. Call for details.

Shorten Development Time, Cut Frustrations BRIEF, The Programmer's Editor

Compile within BRIEF; use autindent; "templates", C specific error support. . . use windows, UNDO, and multiple large files.

But edit YOUR WAY.

Every feature you'd expect and more are integrated elegantly - and it can be modified by you.

You deserve:

"... the best text editor you can buy." - John Dvorak, InfoWorld, 7/8/85

"... the best code editor ..." David Irwin, Data Based Advisor, 8/85
PCDOS \$Call

Make REAL TIME Programming Practical

Csharp Realtime Toolkit

Data acquisition, process control, robotics and devices monitoring applications become practical with Csharp!

Full source code helps tailor programs to various boards and applications.

Reentrant, interrupt handling routines help schedule and react. Fast graphics routines help visualize what is happening.

Control multiple ports reliably, schedule tasks based on events, manage priorities — all with modular, tested, and reliable routines.

Assess and manage the state of hardware at the object level. Let Csharp handle the details.

Portable C source supports RT11 UNIX and MSDOS \$600

Fast File Access with Source Variable Length Fields Save Space

CIndex ISAM Product Line

C-Index contains a high performance ISAM, balanced B + Tree indexing system and variable length fields. The result is a complete data storage system to eliminate tedious programming and add efficient performance to your programs.

Features include random and sequential data access, virtual memory buffering, and multiple key indexes.

With no royalties for programs you distribute, full source code, and variable length fields C-Index/Plus fits what you are likely to need.

Save time and enhance your programs with C-Index/Plus. MSDOS \$349. With C-Index/File for \$89, or/Pro for \$179.

If you call for our advice, you must be completely satisfied with the product you purchase from The Programmer's Shop. If not, you will receive a refund or replacement. Call now for details or our new catalog.

THE PROGRAMMER'S SHOP

128 Rockland Street
Hanover, Massachusetts 02339

800-421-8006

METRIC MINIMIZER

LISTING SEVEN (Listing continued)

```
getnext( string , next )
char **string ;
char **next ;
{
    /*
     * splits the input string "string" into two pieces:
     * "next" contains the first word with no leading or trailing blanks
     * "string" then contains the rest of the line
     */

    int length ;
    char *p ;

    length = strlen( p = *string ) ;
    while( length-- > 0 && isspace( *p++ ) )
        ;

    *next = --p ;

    while( length-- >= 0 && !isspace( *++p ) )
        ;

    *p = '\0' ;
    *string = ++p ;
}
/*****/

check( string )
char *string ;
{
    /*
     * checks if a string starts with a blank
     * if the string was obtained with getnext, this means the string
     * is blank unconditionally report this error
     */

    if ( ! *string )
    {
        printf("Unexpected blank encountered!\n") ;
        return( 1 ) ;
    }
    else
        return( 0 ) ;
}
```

End Listing Seven

LISTING EIGHT

Test input for the VMM program
File name is "al.dat"
Sample input using the Cohen function
Demonstrates using the limit flags and detail debug printout

funname	cohen			
pstart	1.0	1.0		
limitflags	1	1		
lowerlimits	0.5	0.75		
upperlimits	2.0	3.0		
iterlim	10			
debug	2			
next				

This case tests a function which uses "experimental" data
Note the non-default epsilons for this case

funname	sine			
pstart	1.57	-0.65	0.08	-0.005
epsilons	1.0e-141.0e-6			
newdata	b:sine.dat			
iterlim	12			
reset	8			
debug	1			
next				

The following cases use the Rosenbrock function
These correspond to the benchmarks...

Case 1

funnamerosen										
pstart	-1.2	1.0	-1.2	1.0	-1.2	1.0	-1.2	1.0	-1.2	1.0
constants	100.	100.	100.	100.	100.	100.	100.	100.	100.	100.


```

sd
next
Case 2
funnamerosen
pstart -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0
constants 10. 10. 10. 10. 10.
sd
next
Case 3
funnamerosen
pstart -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0
constants 1. 1. 1. 1. 1.
sd
next
Case 4
funnamerosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 100. 100. 100. 100. 100.
sd
next
Case 5
funnamerosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 10. 10. 10. 10. 10.
sd
next
Case 6
funnamerosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 1. 1. 1. 1. 1.
sd
next
Case 7
funnamerosen
pstart -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0
constants 100. 100. 100. 100. 100.
dfp
reset 5
next
Case 8
funnamerosen
pstart -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0
constants 10. 10. 10. 10. 10.
dfp
reset 5
next
Case 9
funnamerosen
pstart -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0
constants 1. 1. 1. 1. 1.
dfp
reset 5
next
Case 10
funnamerosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 100. 100. 100. 100. 100.
dfp
reset 5
next
Case 11
funnamerosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 10. 10. 10. 10. 10.
dfp
reset 5
next
Case 12
funnamerosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 1. 1. 1. 1. 1.
dfp
reset 5
next
Case 13
funnamerosen
pstart -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0
constants 100. 100. 100. 100. 100.
bfgs
reset 5
next

```

(continued on next page)

Instant-C:[™] The Fastest Interpreter for C

**Runs your programs 50
to 500 times faster than
any other C language
interpreter.**

Any C interpreter can save you compile and link time when developing your programs. But only **Instant-C** saves your time by running your program at compiled-code speed.

Fastest Development. A program that runs in one second when compiled with an optimizing compiler runs in two or three seconds with **Instant-C**. Other interpreters will run the same program in two minutes. Or even ten minutes. Don't trade slow compiling and linking for slow testing and debugging. *Only Instant-C will let you edit, test, and debug at the fastest possible speeds.*

Fastest Testing. **Instant-C** immediately executes any C expression, statement, or function call, and display the results. Learn C, or test your programs faster than ever before.

Fastest Debugging. **Instant-C** gives you the best source-level debugger for C. Single-step by source statement, or set any number of conditional break-points throughout your program. Errors always show the source statements involved. Once you find the problem, test the correction in seconds.

Fastest Programming. **Instant-C** can directly generate executable files, supports full K & R standard C, comes with complete library source, and works under PC-DOS, MS-DOS, or CP/M-86. *Instant-C gives you working, well-tested programs faster than any other programming tool.* Satisfaction guaranteed, or your money back in first 31 days. **Instant-C** is \$495.

Rational
Systems, Inc.

P.O. Box 480
Natick, MA 01760
(617) 653-6194

METRIC MINIMIZER

LISTING EIGHT (Listing continued)

```
Case 14
funnamerosen
pstart -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0
constants 10. 10. 10. 10. 10.
bfgs
reset 5
next
Case 15
funnamerosen
pstart -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0 -1.2 1.0
constants 1. 1. 1. 1. 1.
bfgs
reset 5
next
Case 16
funnamerosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 100. 100. 100. 100. 100.
bfgs
reset 5
next
Case 17
funnamerosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 10. 10. 10. 10. 10.
bfgs
reset 5
next
Case 18
funnamerosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 1. 1. 1. 1. 1.
bfgs
reset 5
next
end
ame rosen
pstart 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8 1.2 0.8
constants 1. 1. 1. 1. 1.
bfgs
reset
```

End Listing Eight

LISTING NINE

```
/* Listing 9
 * The way it is done in vmm
 * This works and will be portable
 * so long as addresses are the
 * same size as ints.
 * Fine on 16 bit systems with "small
 * model" e.g. 64K programs
 */

#define          double          float
#include          "fprintf.h"
#include          "scanf.h"
/*-----*/
main()
{
double *(*fun)() ;
double a , b ;

    while(1) {
        readit( &fun , &a , &b ) ;
        doit( fun , a , b ) ;
    }
}
/*-----*/
readit( fun , a , b )
int    *fun ;
double *a ;
double *b ;
{
char    string[10] ;
```



```

scanf("%s %f %f", string, a, b);

funlib( fun, string );
}
/*-----*/
funlib( fun, string )
int *fun;
char string[];
{
double sum();
double mul();

if ( !strcmp( string, "add" ) ) *fun = sum;
else if ( !strcmp( string, "mul" ) ) *fun = mul;
else exit(0);
}
/*-----*/
doit( fun, a, b )
double (*fun)();
double a;
double b;
{
double g;
g = (*fun)( a, b );
printf("%f is the result\n", g );
}
/*-----*/
double sum( a, b )
double a;
double b;
{
printf("%f + %f = %f\n", a, b, a+b );
return( a + b );
}
/*-----*/
double mul( a, b )
double a;
double b;
{
printf("%f * %f = %f\n", a, b, a*b );
return( a * b );
}
}

```

End Listing Nine

LISTING TEN

```

/* Listing 10
 * The correct and most portable way to pass an object which is a
 * pointer to a function
 */

#include <stdio.h>
/*-----*/
main()
{
double (*fun)(); /* fun is a pointer to a function
                  * which returns a double
                  */
double (*readit())(); /* readit is a function that
                       * returns a pointer to a function
                       * that returns a double
                       */

double a, b;

while(1) {
fun = readit( &a, &b );
doit( fun, a, b );
}
}
/*-----*/

double (*readit( a, b ))() /* note position of arguments */
double *a;
double *b;
{
double (*fun)();
double (*funlib())(); /* funlib is a function which
                       * returns a pointer to a function
                       * which returns a double
                       */
}

```

(continued on next page)

Top
Quality
Programming
Tools
from the
Developers
of the
TurboPower
Utilities

NEW!

Turbo EXTENDER™

Tired of fighting 64K Code and Data Segments?
Bored while waiting for your 10,000 liner to Compile?
Want to optimize those sluggish Overlays?

LARGE CODE MODEL

Write Turbo Pascal programs using all 640K of MSDOS memory, based on any number of separately compiled modules. Provides complete parameter passing using normal Pascal syntax. Heap and Data Segment are shared between all modules. No memory-resident kludges or unnatural parameter passing schemes. Comes with a utility which automatically converts your existing applications.

LARGE DATA ARRAYS

Transparently access 1 and 2 dimensional arrays of any conceivable size and type. Four models support Normal RAM to 640K, Expanded memory (EMS) to 2Meg, Virtual (Disk-based) to 30Meg, and sparse arrays like the most advanced spreadsheets. Comes with a fast full-screen array browser.

MAKE FACILITY

A Unix-like MAKE program that is optimized for the Turbo EXTENDER large code model. Rebuild multi-module programs with no wasted effort.

OVERLAY ANALYST

Perform Static and Dynamic analysis of overlayed Turbo programs. Determine sizes of all procedures in each overlay group. Monitor the running program to find the number of overlay reads, procedure calls, and the load address of all procedures.

AND EVEN MORE!

DISK CACHE can be incorporated in your program to speed up disk reads for data bases, overlays, et al. Multi-file full screen BROWSE works on any text file. Pascal ENCRYPTOR makes your source safe from prying eyes, improves compile speed 15-30% and leaves the code 100% functional. SHELL generator creates fast compiling shells of unexercised code.

Two DSDD disks with complete Source Code,
100 page printed manual, 30 day guarantee!
Requires Turbo Pascal 3.0
and DOS 2.X or 3.X. Runs on
IBM PC/XT/AT and compatibles.
Call for generic MSDOS support.

\$85
complete

Also get the TurboPower Utilities
with the acclaimed Pascal Structure Analyzer
Includes a Pretty Printer, Execution Profiler, and
powerful Text and Command Automation Tools.
With full source \$95, executable only \$55.

Credit Card Orders only call Toll-free 7 days per week
(US)800-538-8157x830 (CA)800-672-3470x830
PO, COD, Dealers, Questions, Brochures, call or write:

TURBO
Power

478 W. Hamilton #196
Campbell, CA 95008
ph. 408-378-3672
M-F 9AM-5PM PST

Circle no. 207 on reader service card.

METRIC MINIMIZER

LISTING TEN (Listing continued)

```
char  string[10] ;

scanf("%s %lf %lf", string , a , b ) ;

fun = funlib( string ) ;
return( fun ) ;
}

/*-----*/

double (*funlib( string ))() /* note position of arguments */
char  string[] ;
{
    double (*fun)() ;
    double sum() ;
    double mul() ;

    if ( !strcmp( string, "add" ) )
        fun = sum ;
    else if ( !strcmp( string, "mul" ) )
        fun = mul ;
    else
        exit(0);

    return( fun ) ;
}

/*-----*/
doit( fun , a , b )
double (*fun)() ;
double a ;
double b ;
{
    double g ;
    g = (*fun)( a , b ) ;
    printf("%f is the result\n" , g ) ;
}

/*-----*/
double sum( a , b )
double a ;
double b ;
{
    printf("%f + %f = %f\n" , a , b , a+b ) ;
    return( a + b ) ;
}

/*-----*/
double mul( a , b )
double a ;
double b ;
{
    printf("%f * %f = %f\n" , a , b , a*b ) ;
    return( a * b ) ;
}
```

End Listings

**Z80 CP/M USERS TAKE HEART!
HERE'S ALL YOU NEED TO WRITE
YOUR OWN PROGRAMS FOR ONLY:
\$25.00!**



DR. DOBB'S Z80 TOOLBOOK

By David E. Cortesi

Do you use CP/M? Do you feel as if the only part of the computer industry that has *not* abandoned you is your own Z80 computer? It keeps on working, but when you need programs for it, you have to write

them yourself. When you do, you quickly find that while Pascal or Basic is okay for some things, there is often no substitute for the speed, small size, and flexibility of an assembly language program.

DR. DOBB'S Z80 TOOLBOOK puts the power of assembly language in the hands of anyone who's done a little programming. You'll find:

- **A method of designing programs and coding them in assembly language**—and—a demonstration of the method in the construction of several complete, useful programs.
- **A complete, integrated toolkit of subroutines** for arithmetic, for string-handling, and for total control of the CP/M file system. They bring the ease and power of a compiler's runtime library to your assembly language work, without a compiler's size and sluggish code.

Best of all, every line of the toolkit's source code is there for you to read, and every module's operation

is explained with the clarity and good humor for which Dave Cortesi's writing is known.

Order the Z80 Software on Disk! Save Yourself the Frustration of File Entry

All the software in **DR. DOBB'S Z80 TOOLBOOK**—the programs plus the entire toolkit, both as source code and as object modules for both CP/M 2.2 and CP/M Plus—is yours on disk. (A Z80 microprocessor and the Digital Research RMAC assembler or equivalent are required.)

Receive **DR. DOBB'S Z80 TOOLBOOK**, along with the software on disk, together for only \$40.

To order, return this form with your payment, plus \$1.75 for shipping in the U.S., \$3.75 outside the U.S., to: Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063

For faster service on credit card orders, CALL TOLL-FREE 1-800-528-6050 Ext. 4001. Ask for item #022 for the **Z80 Toolbook** and item #022A for the **Z80 Toolbook** with the disks. Please specify one of the formats offered below

Send me _____ copies of **DR. DOBB'S Z80 TOOLBOOK**:

Send me _____ sets of **DR. DOBB'S Z80 TOOLBOOK** along with the software on disk for \$40 per set:

\$25 ea _____

\$40 ea _____

Subtotal _____

(CA residents add applicable _____% sales tax.)

Shipping _____

TOTAL _____

For the Z-80 software on disk, please specify one of the following formats:

_____ 8" SS/SD _____ Osborne
_____ Apple _____ Kaypro

Check enclosed _____ Charge my _____ Amer. Exp.
_____ VISA _____ M/C

CARD # _____ EXP. DATE _____

SIGNATURE _____

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

PROMPT DELIVERY!

3114C

ANNOUNCING! DR. DOBB'S COMPLETE TOOLBOX OF

*Dr. Dobb's Journal,
the most respected source of
technical software information available,
brings you this collection
of powerful programming tools for C.*

New, from M&T Publishing and
Brady Communications . . .

Dr. Dobb's Toolbook for C

Item #005

A comprehensive library of valuable C code.

Many of Dr. Dobb's most popular articles on C from sold-out issues are updated and reprinted in this unique reference, along with new C programming tools, compilers, line editors, assemblers, text processing programs, and more! Dr. Dobb's Journal offers The Toolbook in a special hardbound edition for only \$29.95. You'll find:

- J.E. Hendrix's famous *Small C Compiler v. 2* and *A New Library for Small C* (also available on disk)
- Never before published: Hendrix's new *Small-Mac: An Assembler for Small C* and *Small Tools: Programs for Text Processing* (both available on disk)
- Ron Cain's original *A Small-C Compiler for the 8080's*
- Plus many useful programming tools in C

Also from M&T Publishing and
Brady Communication—

The Small-C Handbook

Item #006 or #006A

The *Small-C Handbook* by James Hendrix is a valuable resource of information about the Small-C Compiler. In addition to descriptions of the language and the compiler, **The Handbook** also contains the entire source listings of the compiler and its library of arithmetic and logical routines. A perfect companion to the Hendrix *Small-C* compiler offered by Dr. Dobb's on disk, **The Handbook** even tells how to use the compiler to generate new versions of itself. All this is in **The Handbook** for only \$17.95, Item #006; only \$22.95 for **The Handbook** with the **MS/PC DOS Handbook Addendum**, Item #006A.

While both **The Toolbook** and **The Handbook** provide documentation for the Small-C Compiler on disk, **The Handbook** provides more complete documentation and is available with an Addendum for MS/PC DOS versions.



DR. DOBB'S C TOOLS ON DISK

To complement The Toolbook, Dr. Dobbs' also offers the following programs on disk. Source code is included, and except where indicated, both CP/M and MS/PC-DOS versions are available.

Small C Compiler Item #007

Jim Hendrix's Small-C Compiler is the most popular piece of software published in Dr. Dobbs' 10-year history. Like a home study course in compiler design, the Small-C Compiler and The Small-C Handbook provide all you need to learn how compilers are constructed, as well as teaching the C language at its most fundamental level. The Small-C Compiler is available for \$19.95 in both CP/M and MS/PC DOS versions.

Both The Toolbook and The Small-C Handbook provide documentation for the compiler; however, The Handbook provides more complete documentation for both versions, and the MS/PC DOS Small-C Handbook Addendum is recommended in addition to The Handbook for MS/PC DOS specific documentation.

Small-Mac: An Assembler for Small-C Item #012A

Small-Mac is a macro assembler designed to stress simplicity, portability, adaptability, and educational value. The package features simplified macro operators, C-language expression messages, object file visibility, and an externally defined machine instruction table. Included programs are: macro assembler, linkage editor, load-and-go loader, library manager, CPU configuration utility, and dump relocatable files. This program is available with documentation in the Small Mac Manual for \$29.95. For CP/M systems only.

Small Tools: Programs for Text Processing Item #010A

This package consists of programs designed to perform specific functions on text files, including: editing, formatting, sorting, merging, listing, printing, searching, changing, translating, encrypting and decrypting, replacing spaces with tabs and tabs with spaces, counting characters, words, or lines, and selecting printer fonts. Source code only is included. This program is available with documentation in the Small Tools Manual for \$29.95. Both CP/M and MS/PC DOS versions available.

Special Packages—20% Off!

Now, for almost 20% off the individual product prices, you can order a complete set of Dr. Dobbs' C programming tools for your CP/M or MS/PC DOS system!

CP/M C Package Only \$99.95 Item #005A

Ordered individually, these C tools sell for over \$120! Order the CP/M C Package now and you'll receive Dr. Dobbs' Toolbook, The Small-C Handbook, The Small-C Compiler on disk, The Small Tools Text Processor on disk, along with documentation in The Small Tools Manual, The Small-Mac Assembler on disk, and The Small-Mac Manual—all for only \$99.95!

MS/PC DOS C Package Only \$82.95 Item #005B

Individual, these tools sell for over \$100! Order now and you'll receive: Dr. Dobbs' Toolbook, The Small-C Handbook with the MS/PC DOS Addendum, The Small-C Compiler on disk, The Small Tools Text Processor on disk, along with documentation in the Small Tools Manual—all for only \$82.95!

Books	Dr. Dobbs' Toolbook		Check Format		
	Small-C Handbook	Small C Handbook with MS/PC DOS Addendum	CP/M	MS/PC DOS	
Disks	Small-C Compiler (Toolbook or Handbook recommended for documentation)				\$29.95
	Small Tools Text Processor with Small Tools Manual				\$19.95
	Small Mac Assembler with Small Mac Manual (for CP/M only)				\$29.95
					\$99.95
Subtotal					\$82.95
Tax					%
Shipping					TOTAL

CA residents add applicable sales tax
Add \$1.75 per item for shipping in U.S., \$4.25
outside U.S. Add \$8.75 shipping in U.S.
for special C Packages.

For CP/M system disks only, please specify one of the following formats:
☐ Kaypro ☐ Apple ☐ Zenith Z-100 DS/DD ☐ Osborne ☐ 8" SS/SD
Inquire about other formats.

PAYMENT MUST ACCOMPANY YOUR ORDER.
— Check Enclosed
— Charge my ☐ VISA ☐ M/C ☐ Amer. Exp.

Card # _____ Exp. _____
Signature _____ State _____ Zip _____
Name _____
Address _____
(Please use street address)
City _____

To order, return this coupon with your payment to: Dr. Dobbs' Journal,
501 Galveston Dr., Redwood City, CA 94063

PROMPT DELIVERY

For faster service on
credit card orders,
CALL TOLL-FREE 1-
800-528-6050 Ext.
4001. Please refer to
item #'s and specify
desired formats.

LISTING ONE (Text begins on page 116.)

```

SQRT          ; Returns square root of D0 in D0.
              ; Uses algorithm from April 85 EDN

              MOVEM.L D1-D2,-(A7)
              MOVE.L  #40000000H,D2 ; set largest guess
              BRA.S   START          ; enter sizing loop
SIZE          LSR.L   2,D2           ; divide guess by 4
START        CMP.L   D2,D0
              BCS     SIZE           ; branch if guess is greater
              MOVE.L  D2,D1          ; else set root to guess
              SUB.L   D2,D0          ; subtract guess
              LSR.L   2,D2           ; divide guess by 4
LOOP         ; then do normal square root
              SUB.L   D1,D0          ; subtract current root value
              SUB.L   D2,D0          ; and guess
              BMI.S   NEG           ; jump if < 0
              LSR.L   1,D1          ; divide root by 2
              ADD.L   D2,D1          ; add guess to root
              BRA.S   NEXT

NEG          ; if negative then guess
              ; is too large
              ; add guess back in
              ADD.L   D2,D0
              ADD.L   D1,D0
              LSR.L   1,D1          ; divide root by 2
              LSR.L   2,D2          ; divide guess by 4
NEXT         BNE     LOOP           ; done when = 0
              MOVE.L  D1,D0          ; get root
              MOVEM.L (A7)+,D1-D2    ; restore registers
              RTS                ; and end
    
```

End Listing One

LISTING TWO

```

DIV32        ; Divides 32-bit number in D1 by 32-bit number in D0.
              ; Both numbers must be positive, and D0 must be
              ; greater than 65535.
              ; Returns 32-bit quotient in D1, remainder in D0.

              MOVEM.L D2-D3,-(A7)    ; save scratch registers
              MOVEQ   0,D2           ; clear remainder register
              SWAP    D1
              MOVE    D1,D2          ; hi dividend to D2
              CLR.W   D1             ; clear quotient register
              ; D1 low half is quotient, hi half
              ; is low part of dividend
              MOVEQ   15,D3          ; set counter
LOOP         ; divide loop
              ADD.L   D1,D1          ; shift quotient and shift next
              ; bit of dividend to Carry
              ADDX.L  D2,D2          ; shift remainder
              CMP.L   D0,D2          ; remainder < divisor?
              BCC.S   COUNT          ; jump if so, else subtract
              SUB.L   D0,D2          ; divisor from remainder
              ADDQ    1,D1           ; and set bit in quotient
              COUNT  ; decrement counter
              DBRA   D3,LOOP         ; and loop
              MOVE.L  D2,D0          ; remainder to D0
              MOVEM.L (A7)+,D2-D3    ; restore registers
              RTS                ; and end
    
```

End Listing Two

LISTING THREE

```

;
; variation 1, handles arguments 0-255
;
byte_to_dec proc near                ;call with AL = value to convert
;                                  ; DI = addr for string
    add     di,2                    ;point to low digit
    std     ;set for high to low store
    aam     ;convert low order digit
    or      al,'0'                  ;make it ASCII
    stosb   ;and store low digit
    mov     al,ah                   ;load high part
    aam     ;convert high and middle digit
    or      ax,'00'                 ;make them ASCII
    stosb   ;store middle digit
    mov     al,ah
    
```

```

    stosb   ;store high digit
    cld     ;restore direction flag
    ret     ;back to caller
byte_to_dec endp

;
; variation 2, handles arguments 0-99
;
byte_to_dec proc near                ;call with AL = value to convert
;                                  ; DI = addr for string
    aam     ;convert to two digits
    or      ax,'00'                 ;make them ASCII
    xchg    ah,al
    stosb   ;store middle digit
    mov     al,ah
    stosb   ;store high digit
    ret     ;back to caller
byte_to_dec endp
    
```

End Listing Three

LISTING FOUR

```

/*
TAIL.C A utility to dump the last <n> lines of a file
to the Standard Output device (which may
be redirected to a file or printer).

Usage is:      C>TAIL [ -n ] unit:path\filename.ext

Default value for n is 5.

Version 1.0    Nov. 19, 1985
Copyright (C) 1985 Norman McIntosh
May be freely reproduced for non-commercial use.

To compile with Microsoft C:
C>MSC TAIL;
C>LINK TAIL;

*/

#include <stdio.h>
#include <fcntl.h>

#define REC_SIZE 128                /* size of input file records */

main( argc, argv )
int  argc;
char *argv[];
{
    int    handle,                /* handle for input file */
           lines,                /* Number of lines from end
                                to print. */
           bytes_read;

    long   file_ptr,             /* file byte offset, current rec */
           backup(),             /* Function to backup the
                                specified number of
                                records. */

           lseek();

    char   file_buf[ REC_SIZE+1 ]; /* data block from file */

    /* Abort if no filename supplied, or more than 2 parameters. */

    if ( argc < 2 || argc > 3 )
    {
        fprintf( stderr, "\nUsage : tail [-n] <file name>\n" );
        return 1;
    }

    lines = 5;                    /* Set default number of
                                lines. */

    argv++;                        /* Past program name. */

    if ( **argv == '-' )          /* Check for lines option. */
    {
        lines = atoi( *argv + 1 );
        argv++;
    }

    /* Open specified file in raw mode, abort if open fails */

    if ( ( handle = open( *argv, O_BINARY | O_RDWR ) ) == -1 )
    {
        fprintf( stderr, "\ntail: can't find file: %s\n", *argv );
        return 1;
    }

    /* Print filename and number of lines to print. */

    printf( "\nTail of file: %s for %d lines", *argv, lines );
    
```



```

/* Set starting offset into the file. */
file_ptr = backup( handle, lines );

/* Seek to the specified position. */
if ( lseek( handle, file_ptr, 0 ) == -1L )
{ fprintf(stderr, "\ntail: can't seek to end: %s\n", argv);
  return 1;
}

/* Read and print until the end of file. */
while((bytes_read=read( handle, file_buf, REC_SIZE )) >0)
{ /* Force a null on the end of the string. */
  file_buf[ bytes_read ] = '\0';
  printf( "%s", file_buf );
}

close( handle ); /* close the input file. */
return 0; /* return success code */
}

/*
Backup the specified number of records or until the
beginning of the file. Return the offset (a long)
into the file of where that record begins.
*/

static long backup( handle, recs )
int handle, /* Handle to the input file. */
recs; /* Number of records to backup. */
{ long file_ptr, /* Pointer to current record. */
  rslt, /* Return from seek. */
  lseek();
  char file_buf[REC_SIZE], /* data block from file */
  *p; /* Pointer into file_buf. */
  int n; /* Counter. */

  /* Seek to the end of the file. */

  file_ptr = lseek( handle, 0L, 2 );
  if ( file_ptr == -1L )
  { fprintf( stderr, "\ntail: can't seek to file end\n" );
    exit( 1 );
  }

  /* Loop until we have found the specified
  number of records. */

  while( 1 )
  { /* If file_ptr < REC_SIZE then only backup to 0,
    otherwise back up a full REC_SIZE. */

    if ( file_ptr > (long)REC_SIZE )
      n = REC_SIZE;
    else
      n = (int)file_ptr;

    file_ptr -= (long)n; /* Decrement position in file */

    /* Seek to the desired position. */

    rslt = lseek( handle, file_ptr, 0 );
    if ( rslt == -1L )
    { fprintf( stderr, "\ntail: can't seek to file end\n" );
      exit( 1 );
    }

    /* Read the record. */

    if ( read( handle, file_buf, n ) >= 0 )
    { /* Set pointer to end of buffer. */
      p = file_buf;
      p += n;

      /* While we still have characters in the buffer,
      look for a return. If found decrement the number
      records. If the number of records is less than
      zero then exit. */

      while( n-- )
      { if ( *(p--) == '\n' )
        if ( --recs < 0 )
          break;
      }
    }

    /* If we have found the number of records that we
    desire or we are at the beginning of the file then
    increment file_ptr by the number of characters left
    in the buffer. */

    if ( recs < 0 || file_ptr == 0L )
    { file_ptr += n + 1;
      break;
    }
  }

  return file_ptr; /* offset in the file */
}

```

End Listings

You read Dr. Dobb's Journal And You Don't Subscribe?!



Can you afford to miss an issue with information vital to your interests? As a subscriber you can look forward to articles on Small-C, FORTH, MS DOS, S-100, Compiler optimization, Concurrent Programming and more, delivered right to your door. And you'll never miss the issue that covers your project.

**Save over \$13.00 off newsstand prices
for 2 yrs. Save over \$5.00 for 1 yr.**

YES! Sign me up for ___ 2 yrs. \$56.97 ___ 1 yr. \$29.97

- ☐ Check/money order enclosed
- ☐ Charge my Visa, MasterCard
- ☐ American Express
- ☐ Please bill me later

Name _____

Address _____

City _____ State _____ Zip _____

Credit Card _____ Exp. date _____

Account No. _____

Signature _____

Please allow 6-8 weeks for delivery

3053

MAIL TO:

Dr. Dobb's Journal
501 Galveston Dr., Redwood City, CA 94063

16-BIT SOFTWARE TOOLBOX

VDISK Bites Back

I recently had another painful lesson concerning how much faith to put in IBM documentation. My own company's product PC/FORTH, a Forth-83 Standard interpreter/compiler for IBM PCs, PC/ATs, and compatibles, is unlike traditional Forth systems in that it runs as a task under DOS and is carefully integrated into the DOS environment. My coworkers and I spend a lot of time poring over the IBM technical manuals to ensure that we maintain compatibility across all current and future IBM models and to wring the maximum performance out of the video hardware.

In the latest version of PC/FORTH, the video and graphics routines had grown so extensive that we decided to factor them out into a separate resident system driver that would communicate with PC/FORTH through a software interrupt. Because all the IBM technical manuals stated that interrupts 0F0H through 0FFH were not used, we picked interrupt 0FFH for our PC/FORTH message passing and went on to other problems.

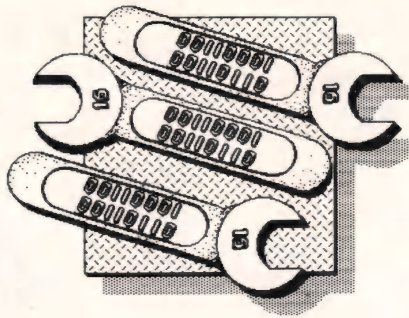
After somewhat more than a thousand copies of this version were in use in the field, we began to get some strange reports of horrible crashes when PC/FORTH was used on a PC/AT with IBM's VDISK installable RAM disk driver. This problem was completely perplexing to us, because PC/FORTH does all of its file-and-record I/O through normal DOS calls, and one logical disk unit looks just like another.

The idea of trying to find the problem in VDISK was intimidating. VDISK is a large chunk of code that is even more obscure than is usual for IBM, and it makes several calls on PC/AT ROM BIOS routines that are both con-

by Ray Duncan

voluted and fragmented.

When an application program running under PC DOS in the 80286's Real Mode accesses a file on the VDISK



RAM disk, the request filters down through the DOS kernel in the usual way. It is translated into logical sector addresses by inspection of the RAM disk's file allocation table and directory and is then passed as logical sector transfer requests to the VDISK device driver.

In order for VDISK to access a logical sector in the VDISK extended memory, it sets up some descriptor tables, stuffs a flag into the CMOS RAM area, and switches the 80286 into Protected Mode. Now, getting back into Real Mode so that VDISK can deliver the data to the requesting program is a bit of a problem. The current models of the 80286 have no provision for returning to Real Mode because it would provide a means for programs to defeat the protection mechanisms altogether.

Instead, VDISK sends a special signal to the keyboard's controller and simply halts the 80286 CPU. The keyboard control recognizes the signal and forces an interrupt of the 80286, which restarts it as though it has just been turned on. The 80286 goes through its usual power-up ROM sequence, fiddles around awhile, and finally inspects the CMOS RAM for the VDISK flag; when it finds the flag, it restores the previous contents of the CPU registers and transfers control back to DOS and the application program.

The fact that this kludge works at all is amazing; the fact that it works fast enough to be useful for anything is a tribute to the power of the 80286.

Sadly enough, after tracing out and marvelling at all this grotesque code,

we were no wiser about why PC/FORTH was crashing when used with VDISK. It took a particularly determined user, armed with a hardware debugging probe and lots of patience, to discover the explanation: VDISK inexplicably trashes the vectors for interrupts 0F0H-0FFH somewhere in the transition from Real Mode to Protected Mode and back again. Even knowing what to look for, we haven't found the little bit of code in VDISK that does the dirty work—but it's in there somewhere!

LaserJet and WordStar

Some months ago I discussed the problems of getting WordStar to work with the Hewlett-Packard LaserJet printer. I eventually obtained a new printer-support disk from MicroPro that gave WordStar control of the LaserJet's italics, boldface, and microjustification, but there was still no support for the proportional fonts needed for really high-quality output.

I have recently been blessed with the following utility programs, from two different companies, that can be used with WordStar document files to produce typeset-quality output on the LaserJet:

- StarJet 2.0 from Control-C Software, 6441 S.W. Canyon Court, Portland, OR 97221; (503) 292-8842. Price: \$150. Requires 45K RAM to execute. Not copy protected.
- Polaris Printmerge from Polaris Software, 310 Via Vera Cruz, Ste. 205, San Marcos, CA 92069; (619) 471-0922. Price: \$99. Requires 196K RAM to execute. Copy protected, but non-copy-protected disk available for an additional \$25.

Polaris Printmerge requires more drastic alterations to your document files and has weaker support for columnar data in the proportional fonts. But both programs are easy to use and allow full access to all the

fonts, proportional spacing, and right justification. They can also draw pretty lines and boxes for generation of custom forms.

Microsoft Macro Assembler

Microsoft has just released a new version (4.0) of its Macro Assembler that contains a number of impressive improvements over previous versions. Overall assembly speed has been improved by a factor of 2 to 3, and the input/output buffers and macro text have been moved out of the symbol space, allowing assembly of larger source files. Most of the bugs mentioned in earlier 16-Bit Toolbox columns have been fixed, including generation of the correct opcodes for all the special 80286 instructions.

A number of new features have been added, including switches to set the file buffer to any size from 1K to 63K, to define a symbol from the command line to control conditional assembly directives, and to check for impure code that would cause problems in 80286 Protected Mode.

The CREF (Cross Reference Utility), LINK (Object File Linker), SYMDEB (Symbolic Debugger), and MAKE (Program Maintenance Utility) utilities have all been enhanced, and two new utilities have been added. These are EXEPACK, which allows you to pack executable files by removing repetitive sequences of bytes, and EXEMOD, which lets you modify the stack size or memory allocation fields of an EXE file header.

The manual for the new assembler and its associated utilities is excellent. Information is well indexed and easy to find. The manual provides virtually no guidance on 8086/80286 programming, however, so you will still need additional reference materials such as Intel's *iAPX 86 User's Guide* or Rector and Alexy's *The 8086 Book*.

68000 Feedback

Lee Robertson of Sandy, Utah, writes: "I found [the 68000 routines] published in the November 1985 16-Bit Toolbox column very interesting and would like to add some suggestions. First of all, Mike Morten suggests using ADD instructions instead of shifts in the square-root routine. I agree this speeds things up, but only by half the amount he states. An

ADD.L instruction requires eight time states instead of six.

"Regarding the random-number routine of November's Listing Four, the following four lines above label DIV2:

```
LSL.L    1,D6
LSL      1,D5
BCC.S    DIV2
ADDQ.L   1,D6
```

can all be replaced by two ADD instructions as follows:

```
ADD      D5,D5
ADDX.L   D6,D6
```

"Not only is this shorter but it is also much faster. These lines are inside the main loop, so any improvements that can be made here will greatly affect the total run time of the program. The loop is taken 15 times, so that the average time savings are 300 time states. Also, the constant 127,773 is used twice inside the main loop. If this is placed in a register instead, it will save another 240

Lattice® Works

RPG COMPILER FOR IBM PC

The new Lattice RPG II compiler is ideally suited for creating commercial applications for MS-DOS. Allow your current RPG II programmers to be productive on MS-DOS.

The Lattice RPG II compiler is compatible with System III, System/34 and /36 RPG II compilers, it uses ASCII files and MS-DOS command language, plus has ISAM compatibility with dBASE III. \$750.00 and no run time fees.

VERSION 3 OF THE LATTICE MS-DOS C COMPILER IS NOW AVAILABLE.

This is a major upgrade of the product and is available to registered users for a \$45 update fee. Non-registered \$60. The list price remains \$500.

New compiler features include:

- *ANSI language constructs...*
 - "unsigned" as a modifier
 - "void" data type
 - "enum" data type
 - structure assignments, arguments, and returns
 - argument type checking
- *Inline code 8087/80287 80186/80286*
- *Code generation*

The compiler also contains numerous improvements such as

better aliasing algorithms, more efficient code generation, and more flexible segmentation. The library includes more than 200 new functions in the following categories:

- *ANSI/UNIX/XENIX compatibility*
- *Extended support for MS-DOS*
- *Extended support for networking, including file sharing, file locking, and I/O redirection*
- *Flexible error handling via user traps and exits*

The Library has also been re-engineered to produce much smaller executables.

LATTICE ANNOUNCES NEW DATA ENCRYPTION SOFTWARE

Now you can keep your confidential data confidential. Thanks to new SecretDisk, a new data encryption system for IBM PC, XT, AT and compatibles.

Utilizing the NBS Data Encryption Standard, SecretDisk provides complete security for salaries, customer lists, or other sensitive information stored on a floppy or hard disk. SecretDisk is loaded as a disk driver by MS-DOS. It creates new DOS drives (like D:) on floppy or hard disks where all data and programs are always fully encrypted.

SecretDisk is extremely easy to use. A password is entered when the system is booted, and protection can be switched on and off with a single password controlled command line. However, without the password, there is no way to access the encrypted files. \$59.95.

Contact Lattice, to discuss your programming needs. Lattice provides C compilers and cross compilers for many environments including Tandy, Sony, Hewlett-Packard, Tandem, and IBM Mainframe. Corporate license agreements available.



Lattice

(312) 858-7950 TWX 910-291-2190
 INTERNATIONAL SALES OFFICES:
 Benelux: De Vooght. (32)-2-720-91-28.
 Japan: Lifeboat Inc. (03) 293-4711
 England: Roundhill. (0672) 54675
 France: SFL (1) 46-66-11-55
 Germany: (49) 7841/4500 (49) 8946/4613-290

NEW!



Dr. Dobb's Journal

BOUND VOLUME 9

All of 1984, OVER 1000 PAGES In One Complete Volume

Bound Volume 9

This volume contains every issue of 1984—the year DDJ got mathematical with articles on the fast Fourier transform, a minimax algorithm, complex numbers, and two encryption systems. It was the year we explored the art of software design with articles on parsing, computing with streams, using pseudocode, and a new column on software design issues. We spoke in many tongues in 1984 with articles on Chinese Forth, a new library for Small-C, an introduction to Modula-2, and a review of Turbo Pascal. And we operated on operating systems with C and our inside Unix issue. (Item #0208)

Vol. 1 1976 (Item # 013)

The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to "finger blistering," front-panel, machine-language programming. This is always pertinent for the bit crunching and byte saving, language design theory, home-brew computing construction include: Tiny BASIC, the (very) first word on CP/M, Topics include: Tiny BASIC, the (very) first word on CP/M, Speech Synthesis, Floating Point Routines, Timer Routines, Building on IMSL.

Vol. 2 1977 (Item #014)

These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of PLOT for microcomputers and a great deal of material centering around the Intel 8080, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly Basic, and NBL. Articles are about Lawrence Livermore Lab's BASIC, Alpha Micro, String Handling, Ciphers, High Speed Interaction, I/O, Tiny Plot & Turtle Graphics, many utilities.

Vol. 3 1978 (Item #015)

This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would follow. Languages covered in depth were SAM76, Plot, Pascal, and Lisp; in addition to RAM Testers, S-100 Bus Standard Proposal, Disassemblers, Editors.

Vol. 4 1979 (Item #016)

This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages. Innovation is still present in every page, and more attention is paid to the best ways to use the processors which

have proven longevity—primarily the 8080/Z80, 6502, and 6800. Subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Pseudo-random Sequences, and Interfacing a Micro to a Mainframe.

Vol. 5 1980 (Item #017)

Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kildall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a subject of greater import, and DDJ published Ron Cain's immediately famous Small-C compiler—a CP/M-Flavored C Interpreter. Ron Cain's C Compiler for the 8080. Further with Tiny BASIC, a Syntax-Oriented Compiler, Writing Language, CP/M to UCSD Pascal File Conversion, Runtime library for the Small-C Compiler.

Vol. 6 1981 (Item #018)

1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, Small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features. Highlights: Information on PCNET, the Conference Tree, and the Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

Vol. 7 1982 (Item #019)

In 1982 we introduced several significant pieces of software, including the RED text editor and the Runic extensible compiler. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi delivered his famous review of JRT Pascal and wrote the first serious technical comparison of CP/M-86 and MSDOS. We looked forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And we looked beyond, in the provocative essay, "Fifth-generation Computers."

Vol. 8 1983 (Item #020)

DDJ turns pro. Some of the most powerful, professional programmer's tools ever published in a magazine are in this volume. The second half of Jim Hendrix's Small C Compiler (continued from Vol. 7), Ed Ream's RED screen editor, A microcomputer subset of the Defense Department's official programming language, Ada, C and Forth and 68000 software.

Buy the complete set
and SAVE 15%
Now, you can have
the entire set of Dr.
Dobb's Journals
from 1976 through
1984, Bound
Volumes 1-9, for
\$225. That's almost
\$40 off the combined
individual prices! To
order the complete
set, Volumes 1-9, ask
for item #020C.

Order Now! Call Toll-Free
1-800-528-6050 Ext. 4001

To order, call toll-free 1-800-528-6050, Ext. 4001. Please refer to book title and item #.

Or, return this coupon to **Software Tools, 501 Galveston Dr., Redwood City, CA 94063.**

YES! Please send me the following Bound Volumes. Payment must accompany your order.

_____ Check/money order enclosed

_____ Please charge my _____ VISA _____ M/C _____ Amer. Exp.

Card # _____ Exp. Date _____

Signature _____

Name _____

City _____ State _____ Zip _____

Vol. 1	\$27.75
Vol. 2	\$27.75
Vol. 3	\$27.75
Vol. 4	\$27.75
Vol. 5	\$27.75
Vol. 6	\$27.75
Vol. 7	\$30.75
Vol. 8	\$34.75
Vol. 9	\$35.75
All 9	\$225

CA residents add applicable sales tax _____ %

Shipping _____

Please add \$2.25 per book in U.S. (\$5.25 each surface mail outside U.S. Airmail rates available on request)

TOTAL _____

time states.

"I have included a square-root routine (Listing One, page 114) that is faster than the routines you previously published. It works on 2 bits at a time, starting with the highest bits in the number. It guesses the square root of 2 bits and keeps testing 2 bits at a time to see if the guess is too large for the number; if so, it divides the guess by 4 to move to the next 2 bits in the number. It repeats this process for each 2-bit combination in a 32-bit number.

"The first loop in the program is used to quickly scale the initial guess down to the size of the input number. This speeds things up greatly for smaller input values. The time required for best case (input=0) is 650 time states; the time increases as the input number gets larger up to a worst case (input=0FFFFFFFH) of 1,322 time states.

"I have also included a listing for a general-purpose 32-bit divide routine. (See Listing Two, page 114.) The method requires that the divisor be greater than 16 bits (65,535). If the divisor is 65,535 or less, the 68000 hardware divide instruction can be used instead, because it will be quite a bit faster even if it has to handle overflow from the divide."

8086 Programming Pearls

Dan Daetwyler, a veteran correspondent of *DDJ*, writes: "Here is a cute method for binary-to-decimal ASCII conversion that might prompt a few readers to exploit the 8086's instruction set more. It assumes that register AL contains the value to be converted and that DI contains the address where the ASCII characters should be stored. The first variation can handle any value between 0 and 255 and always returns three bytes. The second variation assumes an input value between 0 and 99 and always returns two bytes. Many other variations on this theme are possible." Dan's routines are in Listing Three, page 114.

Utility of the Month

The 16-Bit Toolbox just wouldn't be complete this month without a nice MS DOS utility, and Norman McIntosh was kind enough to supply one. His program, named TAIL, works much

like its Unix equivalent and will send the last *n* lines of the designated file to the standard output device (which may be redirected to the printer or a file or piped to another filter program).

The TAIL program is given in Listing Four, page 114. Instructions for use of the program may be found at

the beginning of the source code.

DDJ

(Listings begin on page 114.)

Reader Ballot

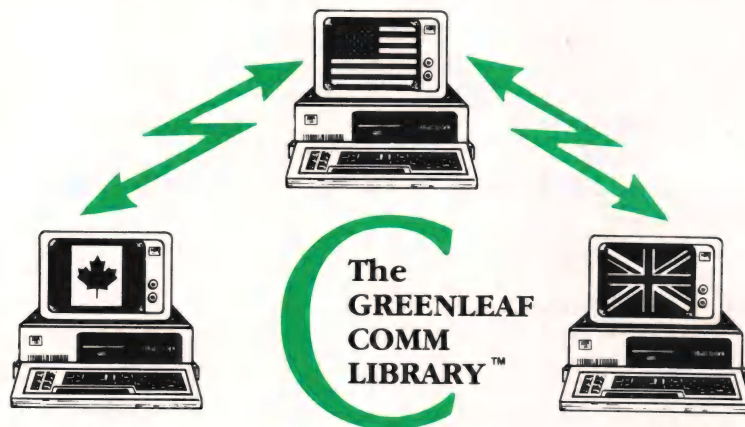
Vote for your favorite feature/article.
Circle Reader Service No. 6.

ARE YOU TRYING TO COMMUNICATE ?

C programs can communicate with the world now through the power of **The Greenleaf Comm Library**. Now from the people who brought you **The Greenleaf Functions** General Library for C, comes this rich interrupt driven, ring-buffered asynchronous communications capability.

Over 100 functions in C and assembler to facilitate communications at up to 9600 baud. Up to eight ports at a time. ASCII or XMODEM. X-On/X-Off too. Hayes compatible modems controlled here. Safe too, bet you can't exit your application with interrupts hot. Major applications around the world base their communicating applications on **The Greenleaf Comm Library**. Stop just trying and start really communicating. Get your copy of **The Greenleaf Comm Library** today. For all major C compilers, all models, all versions. For the IBM PC and just about any machine with MSDOS and an 8086. Comes with source code, extensive examples, demo programs, featuring **C-Terminal**, reference card and newsletter. No royalty. \$185

Other Products: **The Greenleaf Functions** General Library, over 220 functions for total control of the IBM PC, with source. \$185 for the compilers listed below. (See ordering instructions below).



Specify compiler when ordering: Lattice, Microsoft, Computer Innovations, Mark Williams, or DeSmet. Add \$7.00 for UPS Second Day Air (or \$5.00 for ground). Texas residents add sales tax. Mastercard, VISA, check, or P.O. In stock, shipped same day.

<input type="checkbox"/> General Libraries	\$185
<input type="checkbox"/> Comm Library	\$185
<input type="checkbox"/> CI186 Compiler	\$349
<input type="checkbox"/> Lattice C	\$395
<input type="checkbox"/> Mark Williams	\$475

For Information:
214/446-8641

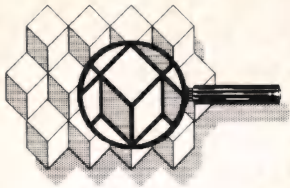
PRICES ARE SUBJECT TO
CHANGE WITHOUT NOTICE.



2101 HICKORY DR.
CARROLLTON, TX 75006

Circle no. 97 on reader service card.

OF INTEREST



Windows

More than half of corporate personal computers will use Microsoft's Windows by 1990, according to David Ferris, chairman of Ferin Corp., a firm that provides PC support for Fortune 1000 companies. He predicts that Microsoft's product will become the industry standard over such competitors as IBM's TopView, Digital Research's GEM, and Quarterdeck's DESQview.

The power of Windows is also expected to increase over the next three to five years for the following reasons: other software developers will adapt their products to it; and hardware will improve to include larger RAM memory, faster processors, and better chip instruction sets.

The latest release of Windows from Syscom Software allows programmers to incorporate on-line help, pop-up windows, and pull-down menus in new or existing applications. Wings, the Window Generation System, lets programmers create windows and generate the source code for them automatically.

C Windows includes modules for all memory models of Computer Innovations, Desmet, Lattice, and Microsoft C. BASIC Windows supports Digital Research CB-86, Microsoft Compiled BASIC, Business BASIC and Quick BASIC, and the IBM BASIC Compiler and Business BASIC Compil-

er. C Windows and BASIC Windows are priced at \$99.95. Wings sells for \$49.95.

Artificial Intelligence

Multibots is a line of electronic-robotic experiment and construction sets from Access Software. The foundation of the product line is a master electronics module that connects to the user's home computer. This module contains all the necessary circuits and components to allow the computer to function as a sophisticated control device that can take measurements and sense temperature, sound, light, and other conditions.

Multibot sets teach the principles and theory of computer-controlled robotics, facilitate experiments and projects in speech and audio digitization and playback, and permit users to take precise electronic measurements with software and hardware that turns the computer into a sensitive digital voltmeter and/or a digital storage oscilloscope. The sets can also sense events and preprogram events. The product line operates on the Commodore 64 and 128 computers. Versions are expected for Amiga, Apple, Atari, and IBM personal computers.

Prolog V-Plus, a Prolog interpreter from Chalcadony Software, features more than 100 predefined predicates and operators, double-precision floating point arithmetic, standard arithmetic functions plus transcendental and trigonometric functions, a large memory model (up to 640K RAM), coresidency (the ability to call other programs), and

addressable cursor and graphics functions. It also adheres to the de facto standard Edinburgh syntax and provides interactive debugging facilities and memory management. Prolog V-Plus requires PC DOS or MS DOS, Version 2.1, and 256K RAM. It is available for \$99.95.

Version 2 of ESP Advisor, from Expert Systems International, allows the construction of knowledge bases of up to approximately 3,000 rules. Prolog-2, as a compiler and interpreter, combines with ESP Advisor. The new version includes the ability to invoke programs written in Prolog during consultations using an optional Prolog-2 interpreter, a virtual knowledge base that allows very large knowledge bases to be created, the referencing of parameters from within any text item displayed, a set of commands, and a full on-line help facility. Version 2 is priced at \$895.

Human Edge Software's Certified Developers' program is sponsoring 20 authors to develop expert system applications using Expert Edge. The Certified Developers' program is designed for users who build an expert system application for resale. Applications under development range from a medical malpractice advisor that will provide physicians with an edge against possible medical malpractice suits to an expert telephone consultant that will advise users on how to reduce business telephone expenses based on a firm's usage, needs, and available alternatives.

Communications

HyperAccess is a software

program from Hilgraeve that runs on PC-compatible computers to allow communications through modems or cable with almost any other computer. It can transfer files at rates of more than 1K per second and allows the computer to act as an unattended host so it can be run from any remote computer or terminal. Other programs, DOS commands, or DOS macros can be used while on-line. HyperAccess is available for use with PC DOS or MS DOS on IBM PC, PC/XT, PC/AT, and PC-compatible or Z-100 computers.

With Polygon's polyShare product, personal computer and VAX/VMS users can build a VAX-based library of personal computer applications. Text or binary file entries can be checked into and out of the polyShare library. The program features entries organized by site-specified category, a menu-interface and automatic PC transfers, multiple libraries, cross-indexing, and free-form library searches. polyShare uses Polygon's poly-XFR file transfer software on the host VAX system and either poly-COM, poly-COM/220, or poly-COM/240 terminal emulation and file transfer products on the personal computer.

A software-based networking system for computers based on the 68000 and 6809 processor families, the OS-9 from Microware Systems, combines the file and input/output systems of all connected computers into one file system. Any network user can access files and I/O devices directly on any other system on the network as if they were local files. The system can be

used with a standard local-area network or long-haul data communications hardware. It is compatible with Ethernet, Omninet, ARCnet, and IEEE-488. Built-in security features are also included.

Based on Motorola 68000 and 6809 microprocessor technologies, the Codex 6740 is a midrange statistical multiplexer that supports interfaces for a mix of asynchronous, synchronous, and bit-oriented protocols. The 6740 nodes support up to eight high-speed composite links ranging in speed from 2,400 bps to 64,000 bps. The system supports up to 19.2K bps and is capable of stat muxing up to 64 channels onto one or more high-speed links. It also provides automatic routing capabilities. The

cost is approximately \$6,000.

Infotron Systems has announced that its InfoStream 1500 high-speed multiplexer is compatible with AT&T's Digital Access and Cross-Connect System (DACS). DACS consists of central-office switching equipment that allows a T1 carrier facility of 1.544M bps to be switched or cross-connected to another T1 carrier.

Utilities

Dogwood Software has announced Helping Hand, a concurrent productivity aid and reference utility for IBM PC, XT, AT, and Type I compatibles. The program runs in a windowing environment allowing users to access a variety of on-line reference

files. Helping Hand requires MS DOS 1.0 or later, 33K minimum RAM, and one floppy disk drive.

Minnow Bear Computers' CBC Manager II is a utility for Digital Research's CBASIC compiler that allows programmers to utilize more memory. It does this by allowing functions to be placed and called from areas normally inaccessible to CBASIC compiler programs. CBC II also has options that allow routines to be called in MS C or MS Pascal. The utility is available for MS DOS and requires MS DOS 2.1 or later to run.

Mastercom, a telecommunications utility from The Software Store, is available for the IBM PCjr, as well as for most computers compatible with IBM PC DOS

and CP/M-80. Mastercom supports Christensen XMODEM, XON/XOFF, line at a time, and no protocol. It features auto-dial, auto-answer, host-mode unattended operation, batch-file transfer, directory display, file erase, file rename, disk drive logging, stored responses, file viewing, upload text throttle, filter or ASCII display options for received control parameters, and menu-installed and menu-driven operation.

Squeezepak, an enhanced disk compression and optimization utility from Demac Software, is intended for use on the DEC PDP-II line of minicomputers using the RSC operating system. A VAX/VMS version is also available. With the system, users can request

C Cross Compiler 68000/08/10/20

Features:

- Full, Standard C
- Easy to Use Compiler Options
- Complete User Documentation
- Global Code Optimization
- Optional Register Allocation Via Coloring
- ROMable and Reentrant Code
- Comprehensive Royalty Free Run-time Library
- Floating Point Library Routines
- Intermix MCC68K C with ASM68K Assembly Language or Microtec PAS68K Pascal
- Optional Assembly Language Listing Intermixed with MCC68K C Source Line Number
- Symbolic Debug Capability

The Microtec MCC68K C Cross Compiler is a complete implementation of the 'C' programming language as defined in The C Programming Language by Kernighan and Ritchie with extensions.

MCC68K emits highly optimized assembly language code for the Microtec ASM68K Motorola compatible assembler.

The Microtec MCC68K package includes the compiler, relocatable macro assembler, linking loader, run-time library, and comprehensive user's guide.

3930 Freedom Circle, Suite 101, Santa Clara, CA 95054
Mailing Address: P.O. Box 60337, Sunnyvale, CA 94088

MICROTEC[®]
RESEARCH

Now Generates:
Position Independent Code

Host computers include: DEC VAX, DG MV-Series, Apollo, IBM PC and PC-compatibles..

We're **Functional** and **Fast** and **Serious** about our products. We've been providing flexible and economical solutions for software developers since 1974.

Beginning with product concept, through development, quality assurance, and post-sales support - **Quality, Compatibility and Service** are the differences which set Microtec Research apart from others.

If you're a serious software developer, shopping for software development tools, call or write today for more information:

800-551-5554,
In CA call (408) 733-2919.

Includes complete source
code and documentation!
Only \$29.95 Each!

Dr. Dobb's Journal of

Software Tools
FOR THE PROFESSIONAL PROGRAMMER

A Unix-like Shell AND a Unix-like Utility Package

by Allen Holub
Both Available on Disk for MS-DOS

THE SHELL Item #180

SH—A Unix-like Shell for MS-DOS

SH is an implementation of the most often used parts of the Unix C shell. This package includes an executable version of the shell along with the complete source code and full documentation.

Supported features are:

- Editing** Command line editing with the cursors is supported. The line is visible as you edit it.
- Aliases** Can be used to change the names of commands or as very fast memory resident batch files.
- History** The ability to execute a previous command again. The command can be edited before being executed.
- Shell variables** Macros that can be used on the command line.
- Nested batch files** A batch file can call another batch file like a subroutine. Control is passed to the second file and then back to the first one when the second file is finished. DOS doesn't have this capability.
- Unix-like syntax** Slash (/) used as a directory separator, minus (-) as a switch designator. A 2048 byte command line is supported. **Command line wild card expansion. Multiple commands on a line.**

The shell also supports redirection of standard input, standard output, and standard error.

This version corrects several bugs found in the original version printed in Dr. Dobb's Journal, December 1985 through March 1986 issues. It runs on any MS-DOS computer.

/util Item #181

A Unix-like Utility Package for MS-DOS

/util is a collection of Unix utility programs for MS-DOS. This package includes complete source code. All programs (and most of the utility subroutines) are fully documented. You'll find executable versions of:

- cat** A file concatenation and viewing program
- cp** A file copy utility
- date** Prints the current time and date
- du** Prints amount of space available and used on a disk
- echo** Echoes its arguments to standard output
- grep** Searches for a pattern defined by a regular expression.
- ls** Gets a sorted directory.
- mkdir** Creates a directory
- mv** Renames a file or directory. Moves files to another directory.
- p** Prints a file, one page at a time.
- pause** Prints a message and waits for a response.
- printenv** Prints all the environment variables.
- rm** Deletes one or more files.
- rmdir** Deletes one or more directories.
- sub** Text substitution utility. Replaces all matches of a regular expression with another string.

**For faster service on credit card orders, CALL TOLL-FREE
1-800-528-6050 Ext. 4001. Please refer to item #'s.**

To order, return this coupon with your payment to: Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063

YES!

Please send me _____ copies of The Shell for \$29.95 each _____
_____ copies of /util for \$29.95 each _____

SUBTOTAL _____

CA residents add applicable sales tax _____%

Please add \$1.75 per disk for shipping _____

TOTAL _____

_____ Check/Money Order enclosed

_____ Please charge my _____ VISA _____ M/C _____ Amer. Exp.

Card # _____ Exp. Date _____

Name _____

Address _____

City _____ State _____ Zip _____

PROMPT DELIVERY!

31140

OF INTEREST

(Continued from page 121)

the generation of statistical profiles describing a disk's fragmentation status throughout the file reorganization process. This information can be displayed on the user's VDT or directed to a disk log file during run time.

Security

Sutton Designs is shipping its sixth-generation surge suppressors under the original ZX-5000 label first introduced in 1981. Twenty-seven models are available in the series, including modem data line protectors and brownout protection models. The sixth-generation introduction includes a five-stage surge and spike system designed to exceed all IEEE 587 1980 Category A test requirements.

Everett Enterprises has released a software encryption program called the Private Line. It conforms to the Data Encryption Standard published by the National Bureau of Standards. MS DOS and CP/M versions are available. The Private Line enables users to send communications (program, data, or text files) over telecommunications networks without fear of interference by third parties.

Rainbow Technologies has released a Xenix version of its Software Sentinel hardware key system that prevents unauthorized access to software programs. The Software Sentinel uses a proprietary encryption algorithm. A series of locks are defined by the software developer and are implemented as part of the program.

Productivity Aids

Lightning is a RAM-resident

disk-caching program from the Portable Computer Support Group (PCSG) that is designed to anticipate what disk accesses are most likely to occur. Algorithms enable the program to build a smart buffer, which keeps the most called-for disk accesses in RAM and is adjustable from 40K to 300K. Lightning is intended to improve the performance of the IBM PC, XT, AT, or any compatible running under MS DOS. A copy-protected version sells for \$49.95; the unprotected version is \$89.95.

The RaceCard-286, a plug-in card that is said to run software up to six times faster than normal with no modification to the software, is available from Mountain Computer. RaceCard emulates the IBM 8088 processor and is compatible with nearly all AT software, RAM, and peripheral cards. The half-card, which measures 5×3.9 inches, uses 7 watts of power from the computer's power supply. The 3Com Ether Series, Novell, Orchid PCnet, and Starlan are among the network packages supported. RaceCard is priced at \$795.

The AT Gizmo is a card from The Software Link that allows PC DOS applications to use 4.6 megabytes of extended memory in the PC/AT. The AT Gizmo with MultiLink Advanced allows up to nine partitions of up to 512K each. It attaches directly to the 80286 processor and operates at up to 10 megahertz. A separate slot in the PC/AT is not required. The product sells for \$295.

Odin Research has developed Otis decision support software to run on any IBM PC or compatible, and on any IBM mainframe or compatible using the CMS operating system. Otis is

compatible with Lotus 1-2-3, Symphony, and dBASE. All data processing takes place in an area of memory that automatically expands to the size of available computer memory. Large groups of items can be handled as a single item using a letter series of names. Processes can be repeated with a built-in language that includes loops and conditional expressions. Otis runs on DOS 2.0 or later and requires 320K minimum memory.

With the ZSTEMpc-4014 from KEA Systems, users can interactively zoom in to display portions of an image that would otherwise be hidden by the limited resolution of the screen. While zoomed in, users can pan to different portions of the image. The zoom and pan functions are controlled by the cursor keys or a mouse and are transparent to the host. Images can be saved and recalled from disk. The system emulates a Tektronix 4014 terminal with the enhanced graphics option. It requires ZSTEMpc-VT100, Version 2.0 or later, and runs on IBM PC, PCjr, XT, AT, or compatible computers with MS DOS, Version 2.0 or later. It also requires 192K of main memory; a color/graphics, enhanced graphics, or Hercules graphics adapter; an internal modem or serial port; and one floppy drive.

Apple

SoftDesign has agreed to act as publisher for MacLightning, a RAM-resident software tool that allows users to access and manipulate a variety of data libraries and reference works. The product is published jointly under an agreement reached between SoftDesign and Target Software. MacLight-

ning, which currently includes a 31,000-word dictionary and a built-in grammar checker, costs \$99.95. It runs with Jazz, Excel, Omnis 3, PageMaker, Word, and ThinkTank. Built-in "Hot Keys" allow users to correct errors within a document, add words to the dictionary, or jump between an application and MacLightning. As words are checked, they are left in RAM to speed up future corrections.

*subCity is a library of subroutines and declarations for Apple Pascal programmers that includes the source texts for more than 100 procedures, functions, and declarations, as well as information on several hard-to-find topics. Topics in *subCity include character, number, and string handling; input prompts; error handling; and search and sort algorithms. A set of routines for disk use provides disk directory lists, disk crunch, date set, file rename, and compare. Assembly-language routines include finding the memory address of any variable, a string finder, and bit-manipulation for character arrays. *subCity is available for Apple II computers with Apple Pascal.

Apple II Pascal, Version 1.3, supports UniDisk 3.5, Apple's recently introduced 3½-inch floppy disk drive that increases floppy disk storage capacity to 800K. It includes the language on both 3½- and 5¼-inch disk formats. The new version runs on any Apple II Plus, IIe, or IIc personal computer with at least 64K of internal memory.

Application Development

Synergy, a TopView-compatible, multitasking oper-

OF INTEREST

(Continued from page 123)

ating environment offering windows, icons, pull-down menus, fonts, and other graphic tools in 12K RAM, allows developers to create user interfaces. Available from Matrix Software, Synergy is compatible with MS DOS or PC DOS (Version 2.0 and later) and TopView. Up to six programs can be run simultaneously as multitasking background processes in windows.

On-Line Software International has announced that DataVantage, a CICS/IMS application testing and development tool, has been enhanced to support the DOS/DLI database environment. DataVantage features ad hoc query, delete, replace, and insert functions in the batch or on-line modes. All accessed data is automatically translated into a display. Segments can be selected for processing based on the contents of one or several fields within the segment.

CompuFirm's Data Base Manager software subsystem enables writers of ap-

plication programs to store, update, and retrieve data records. This can be accomplished on microcomputers functioning under the DOS, iRMX-86, and RTX operating systems. The Data Base Manager maintains data records that are accessed through actions such as open/close database; read/write/re-write/delete record; read next or previous record; and purge or create database. Each format is available for \$495.

StruBAS, the structured BASIC Development System for the IBM PC and compatibles, provides structured programming facilities, full-screen handling, indexed files, and menus. A preprocessor translates BASIC, encased in structured constructs without line numbers, to Microsoft BASIC. Subroutines and record structures for I/O are included, and built-in commands support full-screen and indexed file features. The system, from Laney Systems, also features a file-maintenance program generator, development menus, an ISAM rebuild utility, a source indent util-

ity, and utility subroutines.

CRI's Relate/3000 database management system now operates under Ada. The Application Builder feature serves as an interface between the user, the application being developed, and Relate itself. It uses high-level commands to access Relate commands and constructs, defining screen layouts with page images stored in an application file.

Services

Deltak has released JCL Fundamentals, an eight-course computer-based training series. An introduction to MVS Job Control Language (JCL), this series covers topics ranging from an overview of JCL through common JCL statements, job streams, and debugging JCL errors. The courses deliver instruction on-line using Phoenix/DS.

Reference Map

Access Software, 2561 S. 1560 West, #A, Woods Cross, UT 84087; (801) 298-9077. Reader Service Number 16.

Apple Computer, 20525 Mariani Ave., Cupertino,

CA 95014; (408) 996-1010. Reader Service Number 17. Chaledony Software, 5580 La Jolla Blvd., Ste 126, La Jolla, CA 92037; (619) 483-8513. Reader Service Number 18.

Codex Corp., 20 Cabot Blvd., Mansfield, MA 02048-1193; (617) 364-2000. Reader Service Number 19. CompuFirm, 7677 Ronson Rd., San Diego, CA 92111; (619) 571-0228. Reader Service Number 20.

CRI Inc., 5333 Betsy Ross Dr., Santa Clara, CA 95052; (408) 980-9898. Reader Service Number 21.

Deltak, East/West Technological Center, 1751 Diehl Rd., Naperville, IL 60566; (312) 369-3000. Reader Service Number 22.

Demac Software, 1260 Old Innes Rd., Ottawa, Ont. Canada K1B 3V3; (800) 267-3862. Reader Service Number 23.

dogStar Software, P.O. Box 302, Bloomington, IN 47402; (812) 333-5616. Reader Service Number 24.

Dogwood Software, 1800 Peachtree St., Ste. 510, Atlanta, GA 30309; (404) 355-5272. Reader Service Number 25.

Everett Enterprises, P.O.

TRUE MULTI-TASKING!

TASKVIEW is high tech, available now, and it works with virtually all DOS software. Give Lotus, Sidekick, Multimate or most any DOS program the advantages of real multitasking. It's simple to use, compatible, bulletproof and most of all, it won't slow you down. That's because TASKVIEW only shares your computer when YOU want it shared. At other times, your visible program runs at full speed, waiting for you to easily switch from program to program at the touch of a key. Compatible with most DOS computers including the IBM PC/XT/AT/Jr. series, you can order TASKVIEW today for only \$69.95 + 5.00 Shipping and Handling, VISA and Mastercard.

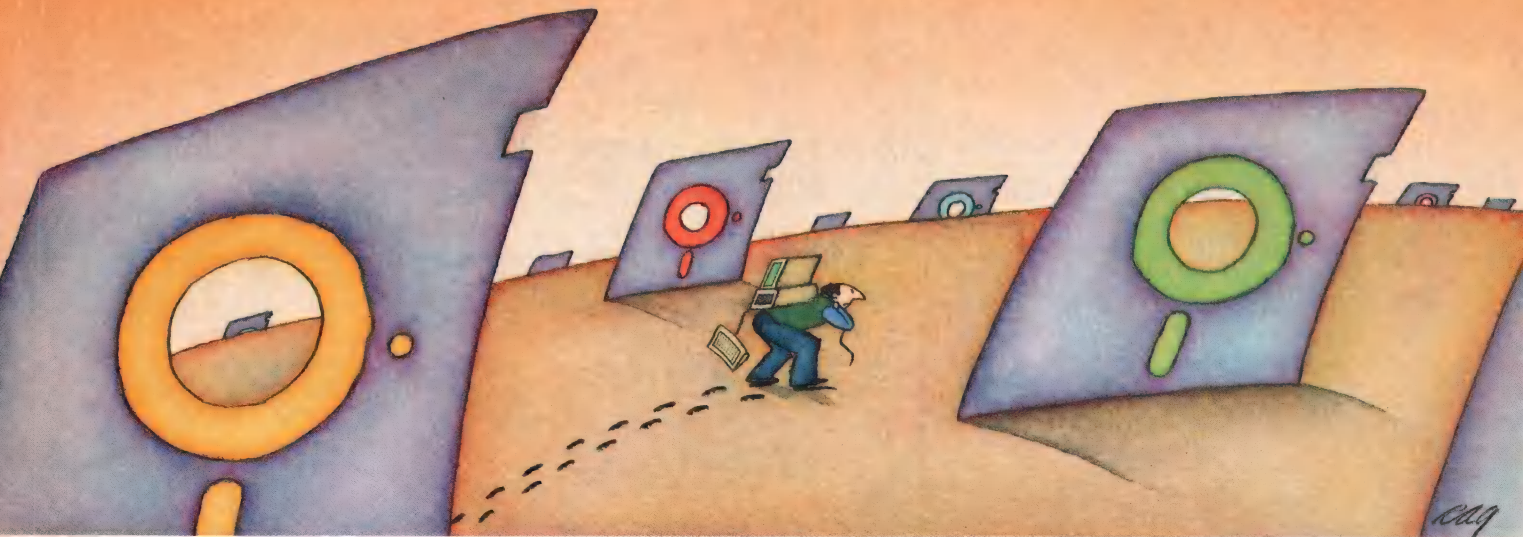
ORDER LINE
(206)367-0650

Lotus trademark Lotus Development Corp.
Sidekick trademark Borland Intl.
Multimate trademark Ashton Tate.

**Sunny Hill
Software**

13732 Midvale North Suite 206
Seattle, Washington 98133





PROBLEM: There's just no easy way to move from one software program to another.

THE SOFTLOGIC SOLUTION: Software Carousel

Now you can keep up to 10 programs loaded and ready to run.

Hard to believe, but some people are happy with just one kind of PC software. Well, this is not a product for them.

But if you're someone who depends on many packages, all the time—someone who'd use several programs at once if you could, well now you can. With Software Carousel.

Why call it "Software Carousel"?

In some ways, Software Carousel works like the slide projector you're used to. You load a handful of pictures, view one at a time, then quickly switch to another. A simple idea, with powerful possibilities for computing.

Here's how it works. When you start Software Carousel, just tell it how much memory you have, load your software and go to work.

Need to crunch numbers? Switch to your spreadsheet. Need your word processor? Don't bother saving your spreadsheet file. Just whip over to your document and do your work. Snap back to your spreadsheet, and it's just like you left it.

With up to ten different programs at your fingertips, you'll have instant access to your database, communications, spelling checker, spreadsheet, word processor, RAM resident utilities, languages, anything you like.

Reach deep into expanded memory.

This could be the best reason ever for owning an expanded memory card, like the Intel Above Board, AST RAMpage, or any

card compatible with the L/L/M Extended Memory Standard.

Software Carousel puts programs into this "high-end" memory for temporary storage when they're not in use. And

switches them back out when you want them. It's fast, efficient, and easy.

If you want, Software Carousel will even use your hard drive for swapping. Just allocate a portion for storage, and go to work.

Sidekick, Superkey and Ready. All at the same time.

You know what happens if you try loading two or more RAM resident utilities at once. You get crashed keyboards, frozen screens, all kinds of interference between programs fighting for control.

With Software Carousel, you can have as many accessories and utilities on-tap as you want. Just load different ones in different Carousel partitions. Since they can't see each other, they can't fight.

The easy way to maximize PC power.

With all this power, you might think Software Carousel is complicated and

difficult to use. Not so. Set it up once, and it will remember forever. Better still, Carousel will look for the programs you use most often, and optimize them for the quickest access.

You can spend a lot more money, and still not get the convenience and productivity increase of Software Carousel.

The way we see it, there are certain things you have the right to expect from your computer. Access to your software is one of them. At just \$49.95 a copy, Software Carousel is the best way to get it. Order today at 1-800-272-9900 (1-603-9900 in NH) or send the coupon below.

Special combination pricing is available for the purchase of Software Carousel and other SoftLogic products, including DoubleDOS and Disk Optimizer.

1-800-272-9900

Software Carousel \$49^{95*}

YES!

Please send me _____ copies of Software Carousel today.

Name _____

Company _____

Address _____

City _____ State/Zip _____

Check Enclosed ☐ VISA ☐ MC ☐ AMEX ☐

Card # _____

Exp. Date _____

Signature _____

SoftLogic Solutions, Inc.
530 Chestnut St.,
Manchester, NH 03101

*plus \$5.00 shipping and handling.

**SOFTLOGIC
SOLUTIONS**

Artificial Intelligence is no longer the exclusive domain of the men in white coats

The Publishers of
**COMPUTER
LANGUAGE**
announce

AI
EXPERT

The first big wave of AI products has swept out of the laboratory and into the marketplace. After 25 years of hope and hype,

commercial applications of AI technology have become a reality. The time has come for an AI programmer's magazine that focuses on real-world applications and problem-solving techniques today!

Become a Charter Subscriber to the first commercial magazine about Artificial Intelligence. . . AI EXPERT. No hype, no star wars nonsense, no pipe dreams. AI EXPERT will focus on practical AI programming methods and applications.

You will learn from the foremost experts about the world of AI:

- Natural Language Processing
- Knowledge Representation
- Machine Vision
- Rule-based Programming
- Heuristic Search Algorithms
- LISP and Prolog programming techniques

- Syntactical Pattern Recognition
- Expert Systems Shell Languages
- And Much, Much More!

Only the publishers of COMPUTER LANGUAGE could produce a quality publication about what's really happening in the AI industry. You'll read real-world application case studies like:

- Intelligent Tutoring Systems
- Medical Diagnosis
- Flight Control Systems
- Resource Planning and Optimization
- Financial Management
- Robotic Assembly
- Computer System Configuration
- Industrial Process Control
- Automatic Programming
- Investment Analysis

We're so sure you'll be more than satisfied with AI EXPERT that we're including an ironclad money-back guarantee. Plus



you'll receive the Premier Issue FREE as a Charter Subscriber. To get your free Premier Issue of AI EXPERT we must receive your subscription coupon by May 15th.

So send in your subscription form today!

Send to: **AI EXPERT**
2443 Fillmore Street
Suite 500
San Francisco, CA 94115

Yes! Start my Charter Subscription to **AI EXPERT**. My 1-year charter subscription is just \$27.00 — 36% discount off the single issue price. By mailing this coupon in by May 15th I also receive the Premier issue of **AI EXPERT FREE!** That's 13 issues for the price of 12.

☐ \$27.00 Payment Enclosed

☐ Bill Me

Name _____

Company _____

Address _____

City _____

State _____

Zip _____

Guarantee: I may cancel my subscription at any time for full refund. Foreign orders must be prepaid in U.S. funds. Please enclose your payment.

Circle no. 286 on reader service card.

Foreign Orders:
Canada \$33.00
World Surface Mail \$39.00
World Air Mail \$57.00

OF INTEREST

(Continued from page 125)

Box 193, Bath, NC 27808; (919) 923-5621. Reader Service Number 26.

Expert Systems International, P.O. Box 53678, William Penn Station, Philadelphia, PA 19105; (201) 337-2300. Reader Service Number 27.

Hilgraeve Inc., P.O. Box 941, Monroe, MI 48161; (313) 243-0576. Reader Service Number 28.

Human Edge Software, 2445 Faber Pl., Palo Alto, CA 94303; (415) 493-1593. Reader Service Number 29. Infotron Systems Corp., Cherry Hill Industrial Center, Cherry Hill, NJ 08003; (609) 424-9400. Reader Service Number 30.

KEA Systems, 2150 W. Broadway, Ste. 412, Van-

couver, B.C., Canada V6K 4L9; (604) 732-7411. Reader Service Number 31.

Laney Systems, 3 Office Park Dr., Ste. 105, Little Rock, AK 72211; (501) 225-7755. Reader Service Number 32.

Matrix Software, 50 Milk St., 15th Floor, Boston, MA 02109; (617) 350-6614. Reader Service Number 33.

Microwave Systems Corp., 1866 N.W. 114th St., Des Moines, IA 50322; (515) 224-1929. Reader Service Number 34.

Minnow Bear Computers, P.O. Box 2233, Station A, Champaign, IL 61820-8233; (217) 398-6883. Reader Service Number 35.

Mountain Computer Inc., 360 El Pueblo Rd., Scotts Valley, CA 95066; (408) 438-6650. Reader Service Number 36.

Odin Research, 834 Old State Rd., Berwyn, PA 19312; (215) 296-4485. Reader Service Number 37.

On-Line Software International, 2 Executive Dr., Fort Lee, NJ 07024; (201) 592-0272. Reader Service Number 38.

Polygon Associates, 1024 Executive Pkwy., St. Louis, MO 63141; (314) 576-7709. Reader Service Number 39. Portable Computer Support Group (PCSG), 11035 Harry Hines Blvd., #207, Dallas, TX 75229; (214) 351-0564. Reader Service Number 40.

Rainbow Technologies, 17971 Skypark Cir., Ste. E, Irvine, CA 92714; (714) 261-0228. Reader Service Number 41.

SoftDesign, P.O. Box 161377, Miami, FL 33116; (305) 253-5521. Reader Ser-

vice Number 42.

Software Link (The), 8601 Dunwoody Pl., Ste. 632, Atlanta, GA 30338; (404) 998-0700. Reader Service Number 43.

Software Store (The), 706 Chippewa Sq., Marquette, MI 49855; (906) 228-7622. Reader Service Number 44. Sutton Designs, 300 N. Tioga St., Ithaca, NY 14850; (607) 277-4301. Reader Service Number 45.

Syscom Software, 320 W. Oak St., El Segundo, CA 90245; (213) 322-2853. Reader Service Number 46.

—Wendelin Colby

DDJ

Productivity Tools from SCE

• **The Seidl Make Utility (SMK)** is the most powerful make utility you can buy for MS-DOS. When changes are made to any program module, SMK will issue only those commands *necessary and sufficient* to rebuild the system. SMK uses a super fast, proprietary dependency analysis algorithm that analyzes *all* dependencies before rebuilding any files. SMK understands complicated dependencies involving nested include files, source, and object code libraries. A high-level dependency definition language makes setting up dependencies easy. It supports parameterized macros, local variables, for loops, constants, include files, command line parameters, in-line and block comments, full pathname and directory support, and more! Works with most compilers, assemblers, and linkers. **\$99.95**

• **The Seidl Version Manager (SVM)** is a state of the art version control system for MS-DOS. It maintains a complete revision history of any text file, without duplication, and can rebuild any previous version of the file. SVM has highly optimized compression routines included for large projects and archiving. Other features include: full pathname and directory support, wildcards, exception cases, on line help, typeset documentation, tutorial, automatic error recovery, and much more. SVM is ideal for all software and text development projects. No other version management system delivers the features, performance, and reliability of SVM. **\$299.95**

• **SMK+SVM** together form an extremely powerful, fully integrated set of productivity tools. **\$379.95**

SEIDL COMPUTER ENGINEERING

1163 E. Ogden Ave., Suite 705-171
Naperville, IL 60540
(312) 983-5477

Circle no. 114 on reader service card.

SUPER TOOLS™

by

PARAGON COURSEWARE

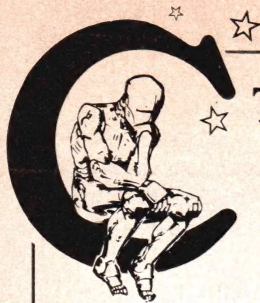
save you time & time & time...

If you program in Turbo™ Pascal this source code Library is a must. Includes **Windowing** package makes pop-up windows and menus easy to program. Allows definition of separate window library. ".....the most general and powerful implementation of overlapping windows available." Computer Language November 1985
Access System & File information from your programs (get dir, change dir, copy files, check hardware etc.) **Evaluate** any **Math Expression** typed by user, fully protected & bomb proof! All with documented source code: **\$75**. Also **Savant Tools™** for your Math programs: Real & Infinite Precision Math. Call for info.

PARAGON COURSEWARE
4954 Sun Valley Rd, Del Mar
California 92014 USA.

(Visa, Master Card OK) Tel (619) 481-1477
Savant, Super Tools: trademarks of Paragon Courseware. Turbo Pascal: trademark of Borland International.

Circle no. 281 on reader service card.



Thinking about C?

**Stop Thinking-
Start Programming Today!**

SPECIAL INTRODUCTORY OFFER!
C' Prime, **Personal Computing and C**,
Plus Apprentice C.
A \$169 value only **\$99**

NEW FROM MANX AZTEC!

C' Prime ~~\$99~~ \$79

Never has C been easier to learn. **Manx Aztec** is now offering a complete C system called **C' Prime** at an exceptionally low price. This powerful system includes a Compiler, Linker, Assembler, Editor, Libraries and Object Librarian. **C PRIME** supports a host of third-party software.

C Apprentice ~~\$49.95~~ \$39.95

Learn C quickly with this complete, easy-to-use C language interpreter. Apprentice C includes a complete one-step compiler that executes with lightning speed, an editor, and a run-time system.

NEW FROM ASHTON-TATE!

Personal Computing and C

A detailed, easy-to-understand guide to C programming prepared especially by Ashton-Tate for use with the new Aztec C' Prime. Includes chapters on C programming basics, function libraries, data handling, and advanced features, plus a complete money management demonstration program.

To order or for information call:

TECWARE
1-800-TEC-WARE

(In NJ call 201-530-6307)



UNIX is a registered TM of Bell Laboratories. dBase TM Ashton-Tate, Inc. MANX AZTEC, C PRIME, Apprentice C TM Manx Software Systems, Inc.

Circle no. 223 on reader service card.

ADVERTISER INDEX

Reader Service No.	Advertiser	Page No.	Reader Service No.	Advertiser	Page No.
221	Alcyon Corp.	61	223	Manx Software	128
121	Arity Corp.	15	109	Manx Software	85
242	Ashton-Tate	27	222	Manx Software	79
248	Ashton-Tate	16	108	Manx Software	7
216	Atron	54	102	Mark Williams	11
250	Austin Code Works	95	110	Micro Interface Corp.	40
123	BD Software, Inc.	44	*	Micomint, Inc.	49
159	Blaise Computing	2	105	Microprocessors Unlimited	87
161	Borland International	64-65	*	Microtec Research	121
202	Brady Computer Books	93	261	Miken Optical Co.	40
219	Brady Computer Books	39	*	Mix Software	19
285	Brady Computer Books	72	128	Morgan Computing	95
267	Brown Bag Software	69	191	Northwest Computer Algorithms	34
181	C Users Group	84	243	Norton Utilities (The)	57
*	C Ware	99	251	Nostradamus	22
226	Cauzin Systems, Inc.	4,5	124	Optotech	1
178	Chalcedony	17	192	Overland Data Inc.	37
81	Cogitate, Inc.	84	281	Paragon Courseware	127
272	Communications Research Group	99	76	Personal Tex	79
237	CompuServe	20	91	Phoenix Computer Products	C2
237	CompuServe	21	139	Phoenix Computer Products	71
286	Computer Language	126	169	Poor Person Software	37
122	CompuView	33	*	Precise Electronics	74
225	Computel Publishing Society	81	143	Programmer's Shop	82-83
96	Computer Innovations	C3	141	Programmer's Shop	105
282	Cosmos	74	107	Quilt Computing	86
283	Cosmos	75	145	Rational Systems	107
82	Creative Programming	53	213	68 Micros	80
268	Custom Software Systems	103	*	SAS Institute Inc.	29
263	Data Management Consultants	13	114	Seidl Computer Engineering	127
203	Datalight	41	85	Semi Disk Systems	60
258	Desktop AI	75	83	Soft Advances	57
87	Digital Research Computers	97	259	Soft Focus	35
*	DDJ Allen Holub-Shell	122	284	SoftLogic Solutions	125
*	DDJ Back Issues	61	262	Software Horizons Inc.	89
*	DDJ Bound Volume	118	153	Solution Systems	56
*	DDJ C Products	112-113	155	Solution Systems	56
*	DDJ Sourcebook	77	147	Solution Systems	101
*	DDJ Subscription	115,124	148	Solution Systems	51
*	DDJ Z-80 Toolbook	111	152	Solution Systems	51
179	Earth Computers	58	240	Southern Pacific Computer Pdts.	30
253	Eclipse Systems	87	164	Spruce Technologies	91
89	Ecosoft, Inc.	45	172	Sunny Hill Software	124
90	Edward K. Ream	81	173	TLM Systems	57
138	Essential Software	48	174	TLM Systems	59
93	FairCom	79	175	TLM Systems	61
*	Gimple Software	47	230	TSF	80
97	Greenleaf Software	119	245/	Tech PC	43
132	Harvard Softworks	71	279		
274	Hauppauge Computer Works	C4	231	Terra Base Software	93
278	ICD	88	234	Think Technology, Inc.	23
194	InfoPro Systems	73	207	Turbo Power Software	109
*	Integral Quality	31	77	UniPress Software	73
100	Kriya Systems, Inc.	67	112	Wendin, Inc.	9
186	Lahey Computer Systems	103	269	Western Wares	87
101	Lattice, Inc.	117	280	Windowsoft	84
252	Levien Instrument Co.	73	116	Wizard Systems	69
117	Librarian (The)	41	244	Workman & Associates	91
257	Logitech, Inc.	63			

*This advertiser prefers to be contacted directly: see ad for phone number.

Advertising Sales Offices

East Coast
Walter Andrzejewski (617) 868-1524

Midwest
Michele Beaty (317) 875-8093

Northern California/Northwest
Lisa Boudreau (415) 366-3600

Southern California/AZ/NM/TX
Michael Wiener (415) 366-3600

Advertising Director
Shawn Horst (415) 366-3600

"The C86 C Compiler is Great...

Computer Innovations' Support is Even GREATER"

DALE HILLMAN,
PRESIDENT, XOR CORPORATION
CREATOR OF "NFL CHALLENGE"



When Dale Hillman decided to create the most exciting football simulation game ever, he knew he needed good language support. The portability and maintainability of C made it a natural choice. **Which** C compiler to choose was another matter entirely.

"Of the many C compilers available, choosing the best one for the job was not easy. Comparing benchmarks, most compilers were strong in one or two categories, yet decidedly weak in others.

Computer Innovations' C86 was the exception. I found the C86 Compiler consistently strong in all categories.

"C86 had a reputation for being a solid, reliable, high-performance compiler. 8087 math support, source level debugging — it had it all. BEST of all was Computer Innovations' incredible technical support. Their highly knowledgeable support team was always available. Their assistance helped cut development

time substantially. And since NFL CHALLENGE took 12 1/2 man-years to create — every little bit helped. It was a service you just can't place a dollar value on..."

If you're working on the next great program, call Computer Innovations. We'll show you why you'll never have to look any further than C86.

**For Further Details
Call Toll-Free:
800-922-0169**

Behind Innovative Programs — Computer Innovations

 **COMPUTER
INNOVATIONS, INC.**

980 Shrewsbury Avenue,
Tinton Falls, NJ 07724 USA (201) 542-5920

EUROPEAN DISTRIBUTOR
Boston Micro, Inc., TELEX: 6712477 BMI USA

©1986 Computer Innovations, Inc.
* NFL Challenge is a trademark of NFL Properties

HAUPPAUGE

Is Getting A Fast Reputation.



HAUPPAUGE started earning a fast reputation with their 87 Math Pak, the combination of an 87 chip and 87 Software Pak that's been accelerating PC math since 1982.

Next came their racy 287 FAST/5, a math coprocessor module with its own 5MHz clock, speeding up PC/AT math by 25%. (Pictured above.)

Now, Hauppauge Unveils the 287 FAST/8A...

Our newest math coprocessor for the PC/AT, the 287 FAST/8A moves out at 8MHz—doubling the speed of each floating point math operation. The FAST/8A accelerates AutoCad, 1-2-3, Symphony, Turbo Pascal, Framework and more. The FAST/8A also runs in PC/AT compatibles including the Compaq Deskpro 286, Sperry PC/IT and TI Business-Pro computers.

...And the 87 Software Pak Version 6.0

Designed to steal the heart of programmers, the 87 Software Pak supports IBM's BASIC Compiler 1.0 and 2.0, and Microsoft's QuickBASIC, executing math-intensive programs up to 20 times faster! The 87 Software Pak also performs FFT's and Matrix operations. For example, a PC (or PC/XT) with an 87 Chip and 87 Software Pak can perform a 512-point complex FFT in just 1.1 seconds. What's more, a PC/AT with a FAST/8A inverts a 25 by 25 element matrix in under 1 second.

Circle no. 274 on reader service card.

Hauppauge

(Pronounced "Ha-pog")

HAUPPAUGE Math Coprocessors

287 FAST/8A 8MHz math coprocessor for PC/AT and compatibles\$379

287 FAST/5 5MHz math coprocessor for PC/AT and compatibles.....\$249

287 Chip PC/AT math coprocessor—runs at 4MHz in PC/AT.....\$219

87 Chip Math coprocessor for IBM PC, PC/XT and compatibles\$129

87-2 Chip Math coprocessor for 8MHz PC compatibles...\$195

HAUPPAUGE Math Coprocessor Paks

87 Math Pak V.6.0 87 chip and math coprocessor software support for IBM BASIC Compiler 1.0, 2.0 and Microsoft's QuickBASIC. Plus, Matrix and FFT support, one year of free updates, complete source code and "8087 Applications and Programming"\$279

87 Software Pak V.6.0 Math coprocessor software support as in the 87 Math Pak, but without 87 chip.....\$180

With any Hauppauge math coprocessor\$150

Recalc + Math coprocessor support for 1-2-3 version 1A...\$ 95

With any Hauppauge math coprocessor\$ 49

HFT + Complete Hayes Fourier Transform Package\$125

With any Hauppauge math coprocessor\$ 79

The 287 FAST/8 Doubles Your PC/AT's Math Speed!

Help your PC/AT get a fast reputation with Hauppauge's new 287 FAST/8A. Call today, or contact your local computer dealer to learn more about Hauppauge's racy product line. And ask for "87 Q & A," our free booklet on math coprocessors.

Hauppauge Computer Works, Inc.

358 Veterans Memorial Highway, Suite MSI,
Commack, New York, USA 11725 • 516-360-3827
Available at your local computer dealer